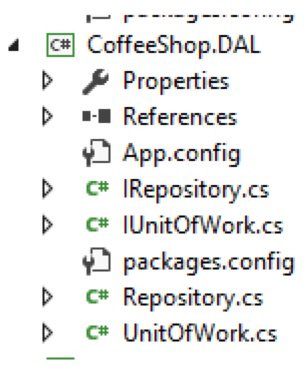
BÀI THỰC HÀNH SỐ 10 (4 tiết)

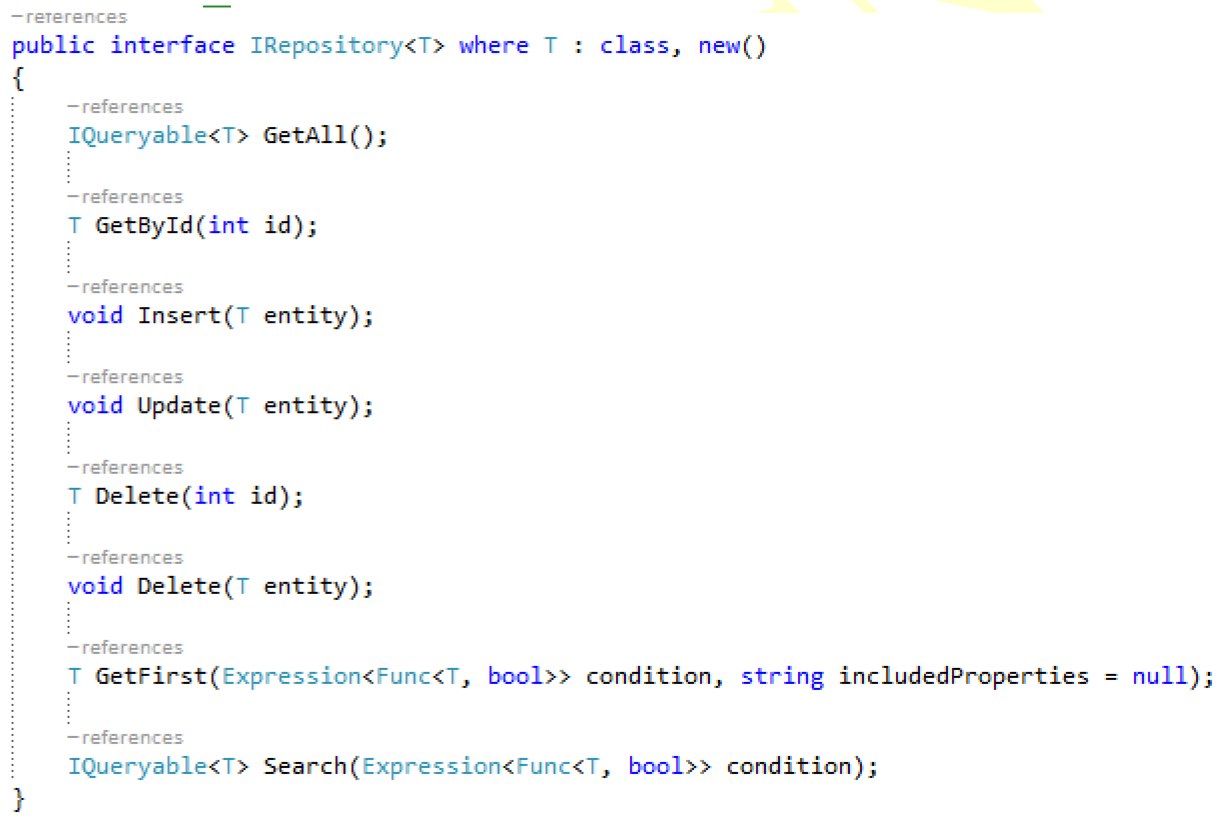
REPOSITORY AND UNIT OF WORK PATTERN

I. Mục tiêu:

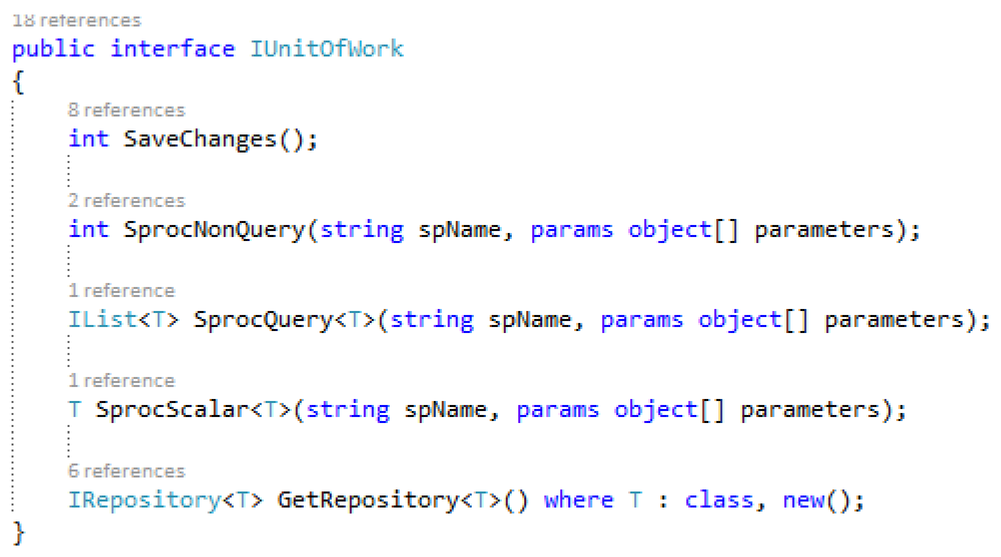
Minh họa cho sinh viên cách sử dụng mẫu thiết kế Repository và Unit Of Work khi xây dựng ứng dụng. Ứng dụng này sử dụng Entity Framework.

II. Thực hành:

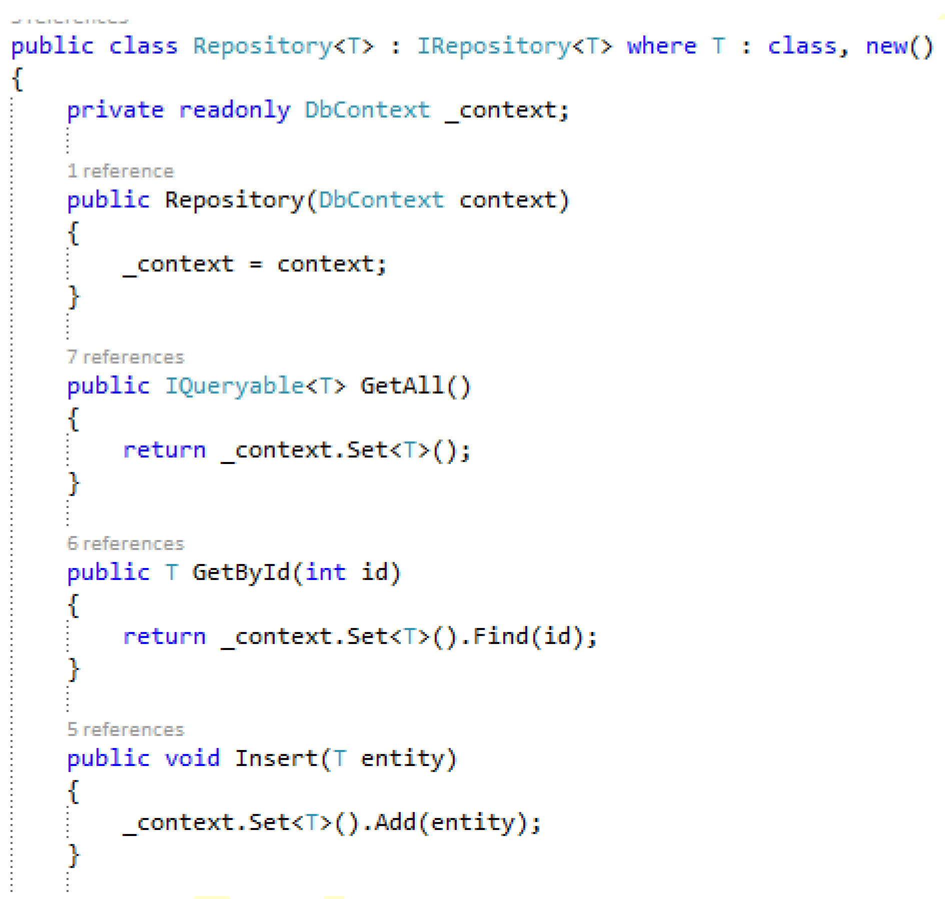




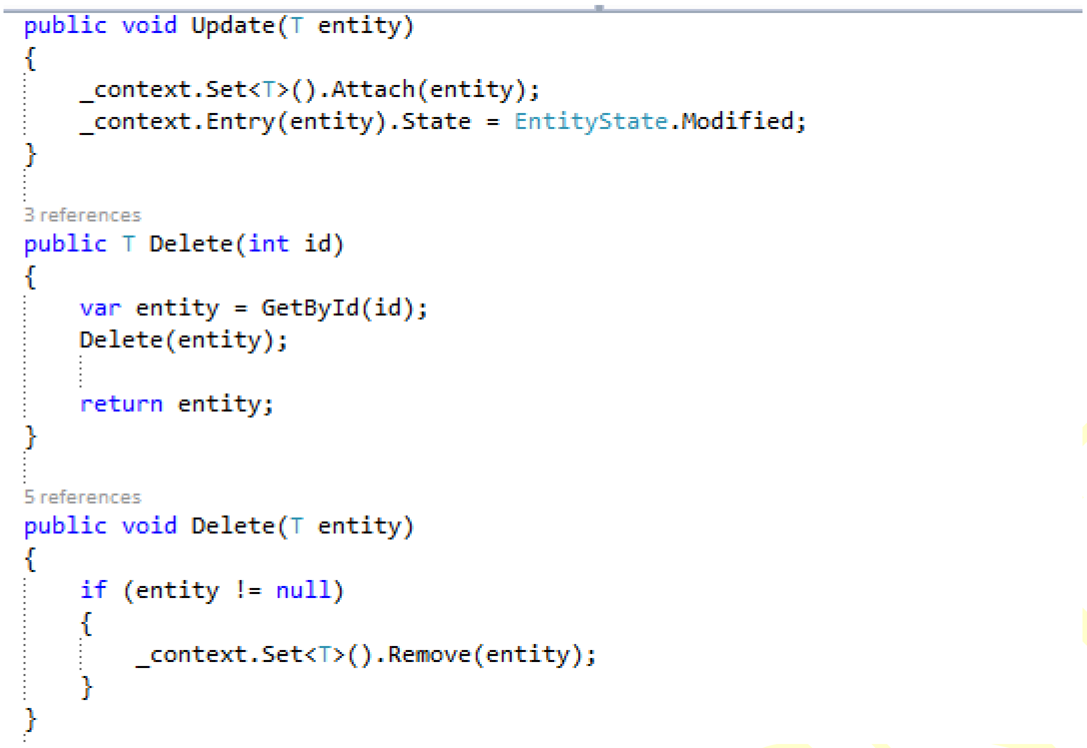
```
public interface IRepository<T> where T : class, new()
{
    IQueryable<T> GetAll();

    T GetById(int id);

    void Insert(T entity);

    void Update(T entity);

    T Delete(int id);

    void Delete(T entity);

    T GetFirst(Expression<Func<T, bool>> condition, string includedProperties = null);

    IQueryable<T> Search(Expression<Func<T, bool>> condition);
}
```



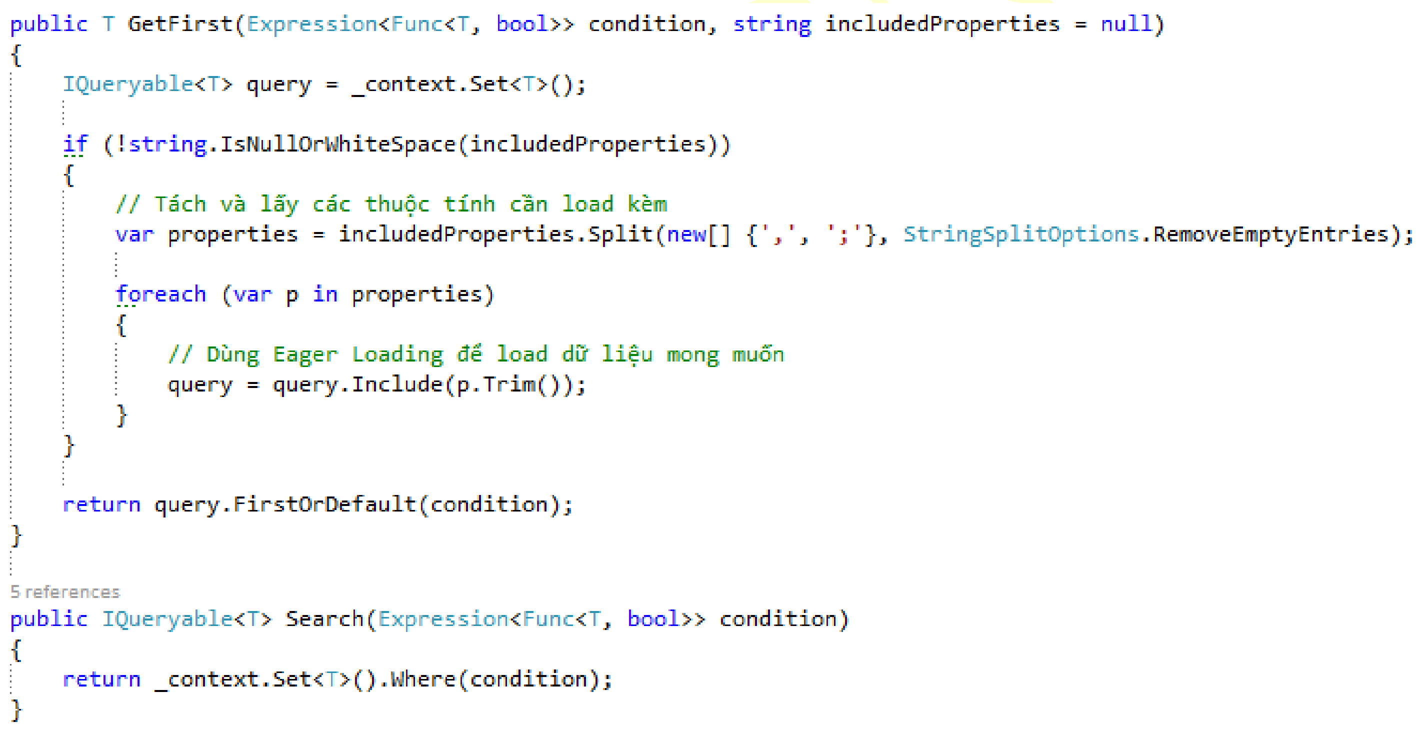
```
18 references
public interface IUnitOfWork
{
    8 references
    int SaveChanges();

    2 references
    int SprocNonQuery(string spName, params object[] parameters);

    1 reference
    IList<T> SprocQuery<T>(string spName, params object[] parameters);

    1 reference
    T SprocScalar<T>(string spName, params object[] parameters);

    6 references
    IRepository<T> GetRepository<T>() where T : class, new();
}
```



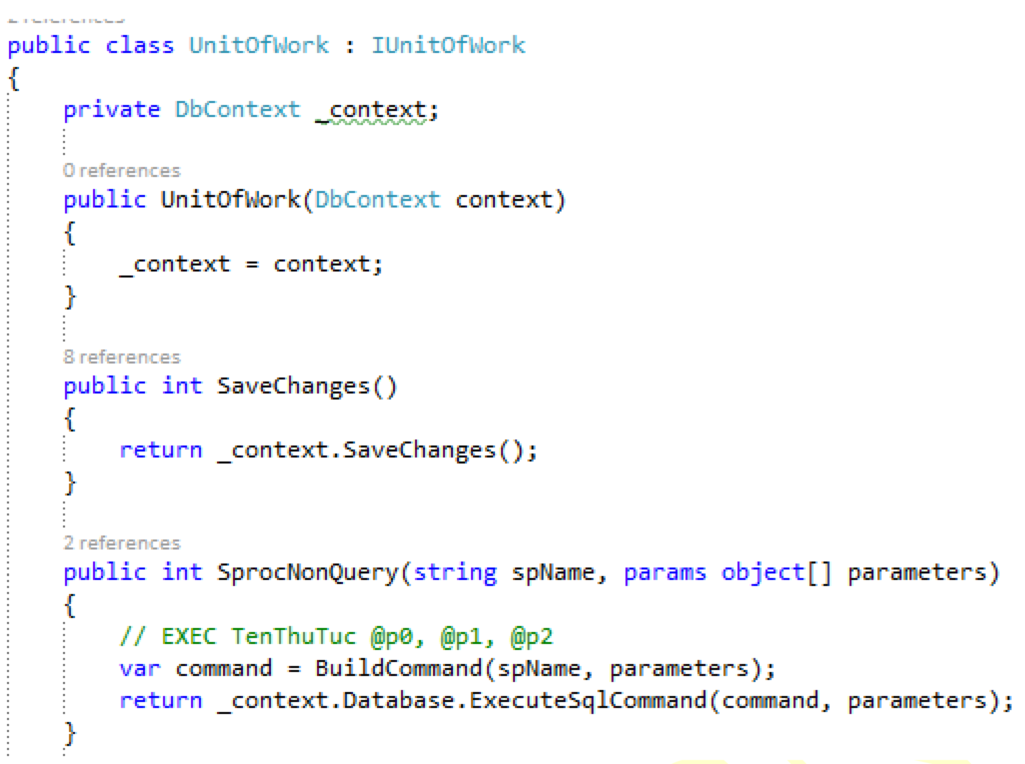
```
public class Repository<T> : IRepository<T> where T : class, new()
{
    private readonly DbContext _context;

    1 reference
    public Repository(DbContext context)
    {
        _context = context;
    }

    7 references
    public IQueryable<T> GetAll()
    {
        return _context.Set<T>();
    }

    6 references
    public T GetById(int id)
    {
        return _context.Set<T>().Find(id);
    }

    5 references
    public void Insert(T entity)
    {
        _context.Set<T>().Add(entity);
    }
```



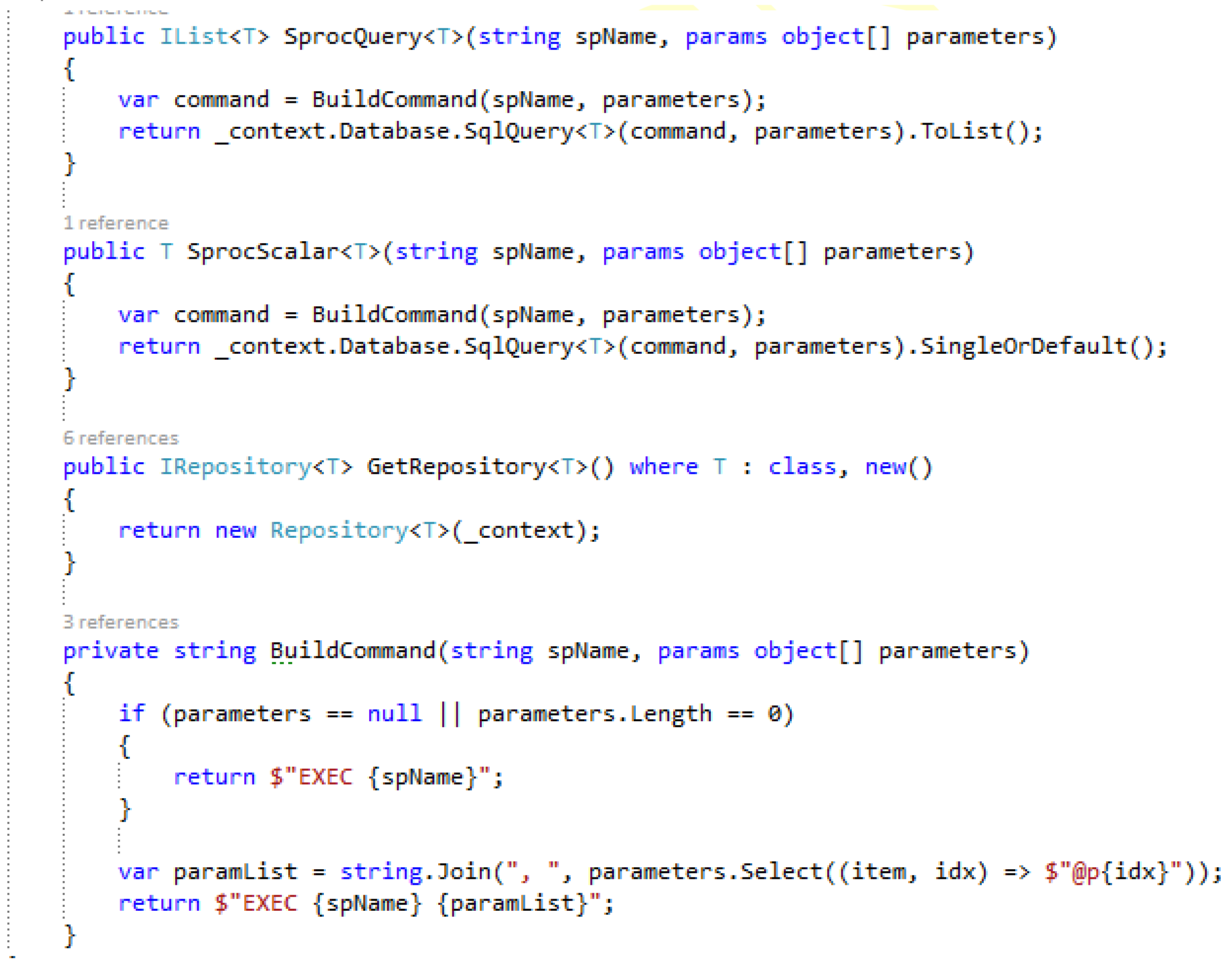
```
public void Update(T entity)
{
    _context.Set<T>().Attach(entity);
    _context.Entry(entity).State = EntityState.Modified;
}

3 references
public T Delete(int id)
{
    var entity = GetById(id);
    Delete(entity);

    return entity;
}

5 references
public void Delete(T entity)
{
    if (entity != null)
    {
        _context.Set<T>().Remove(entity);
    }
}
```



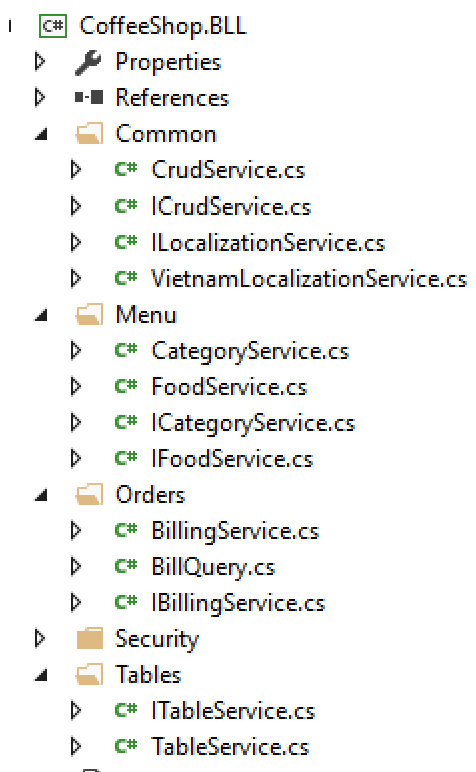
```
public T GetFirst(Expression<Func<T, bool>> condition, string includedProperties = null)
{
    IQueryable<T> query = _context.Set<T>();

    if (!string.IsNullOrWhiteSpace(includedProperties))
    {
        // Tách và lấy các thuộc tính cần load kèm
        var properties = includedProperties.Split(new[] {',', ';'}, StringSplitOptions.RemoveEmptyEntries);

        foreach (var p in properties)
        {
            // Dùng Eager Loading để load dữ liệu mong muốn
            query = query.Include(p.Trim());
        }
    }

    return query.FirstOrDefault(condition);
}

5 references
public IQueryable<T> Search(Expression<Func<T, bool>> condition)
{
    return _context.Set<T>().Where(condition);
}
```



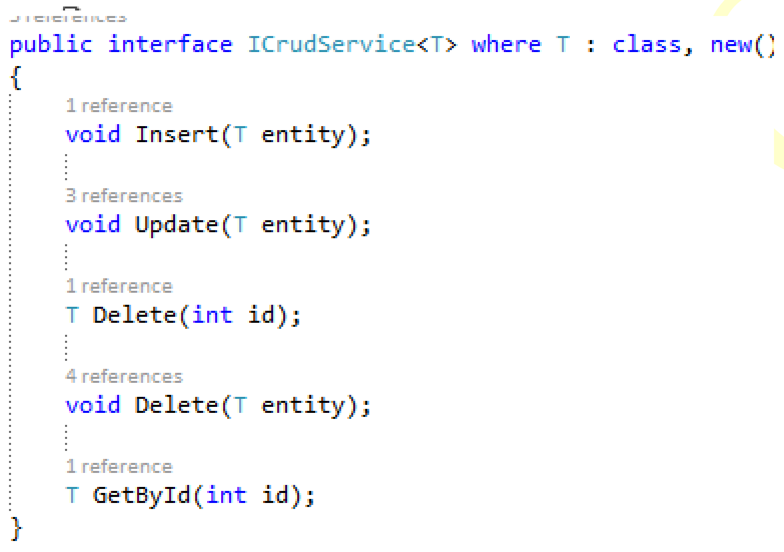
```
public class UnitOfWork : IUnitOfWork
{
    private DbContext _context;

    0 references
    public UnitOfWork(DbContext context)
    {
        _context = context;
    }

    8 references
    public int SaveChanges()
    {
        return _context.SaveChanges();
    }

    2 references
    public int SprocNonQuery(string spName, params object[] parameters)
    {
        // EXEC TenThuTuc @p0, @p1, @p2
        var command = BuildCommand(spName, parameters);
        return _context.Database.ExecuteSqlCommand(command, parameters);
    }
```



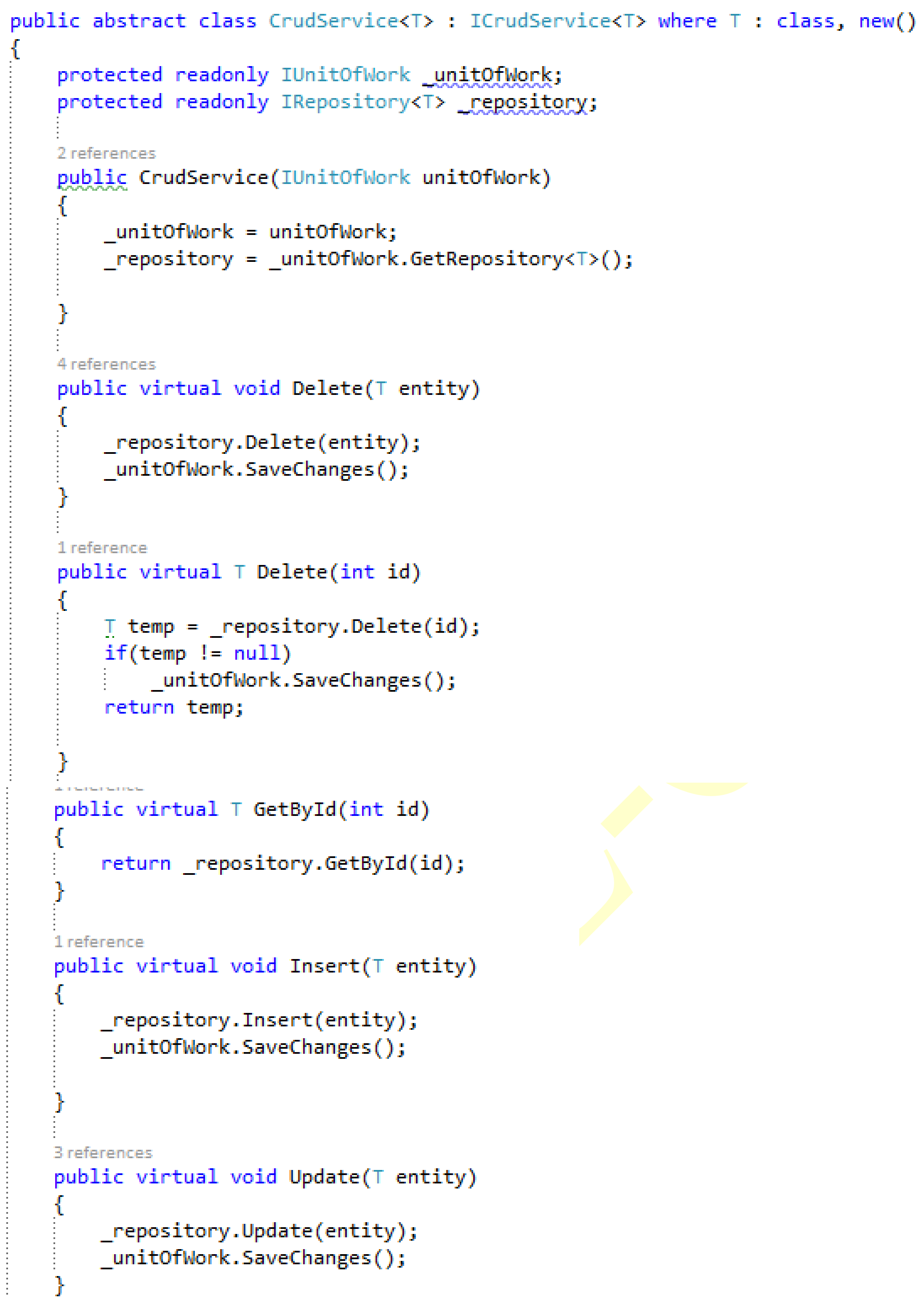
```
    public IList<T> SprocQuery<T>(string spName, params object[] parameters)
    {
        var command = BuildCommand(spName, parameters);
        return _context.Database.SqlQuery<T>(command, parameters).ToList();
    }

    1 reference
    public T SprocScalar<T>(string spName, params object[] parameters)
    {
        var command = BuildCommand(spName, parameters);
        return _context.Database.SqlQuery<T>(command, parameters).SingleOrDefault();
    }

    6 references
    public IRepository<T> GetRepository<T>() where T : class, new()
    {
        return new Repository<T>(_context);
    }

    3 references
    private string BuildCommand(string spName, params object[] parameters)
    {
        if (parameters == null || parameters.Length == 0)
        {
            return $"EXEC {spName}";
        }

        var paramList = string.Join(", ", parameters.Select((item, idx) => $"@p{idx}"));
        return $"EXEC {spName} {paramList}";
    }
```



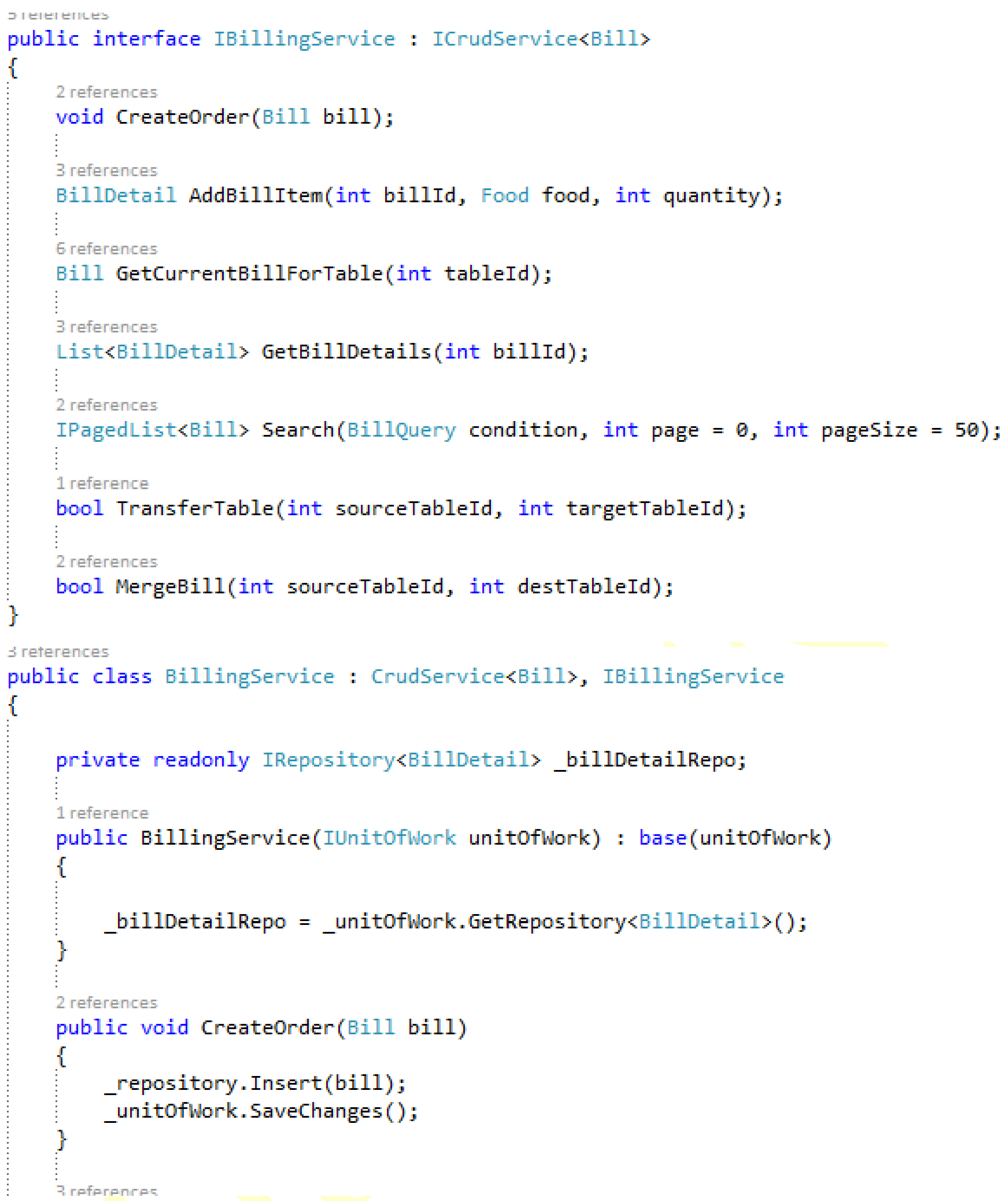
```
CoffeeShop.BLL
  ▷ Properties
  ▷ References
  ◢ Common
      ▷ CrudService.cs
      ▷ ICrudService.cs
      ▷ ILocalizationService.cs
      ▷ VietnamLocalizationService.cs
  ◢ Menu
      ▷ CategoryService.cs
      ▷ FoodService.cs
      ▷ ICategoryService.cs
      ▷ IFoodService.cs
  ◢ Orders
      ▷ BillingService.cs
      ▷ BillQuery.cs
      ▷ IBillingService.cs
  ▷ Security
  ◢ Tables
      ▷ ITableService.cs
      ▷ TableService.cs
```



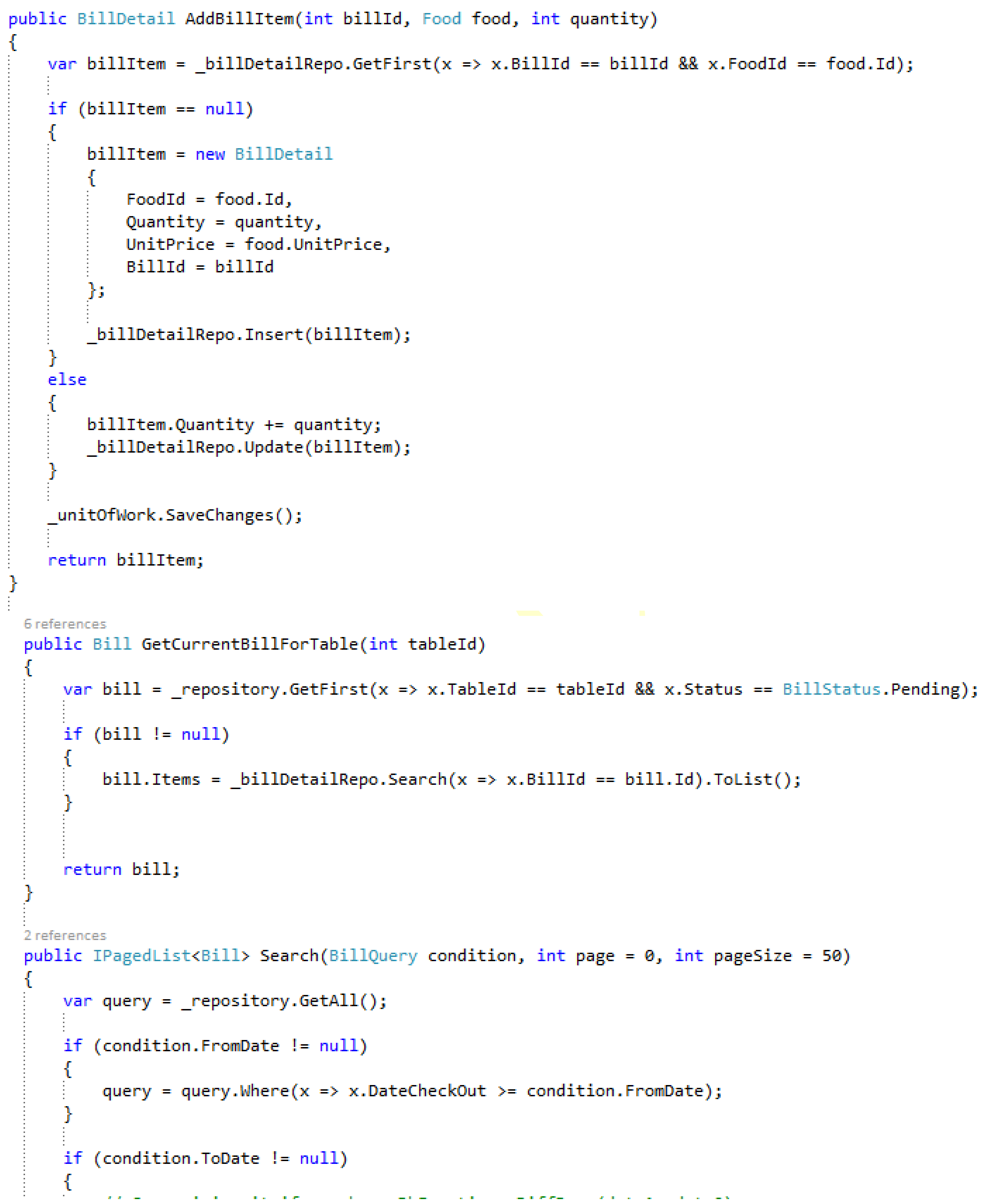
```csharp
public interface ICrudService<T> where T : class, new()
{
    1 reference
    void Insert(T entity);

    3 references
    void Update(T entity);

    1 reference
    T Delete(int id);

    4 references
    void Delete(T entity);

    1 reference
    T GetById(int id);
}
```



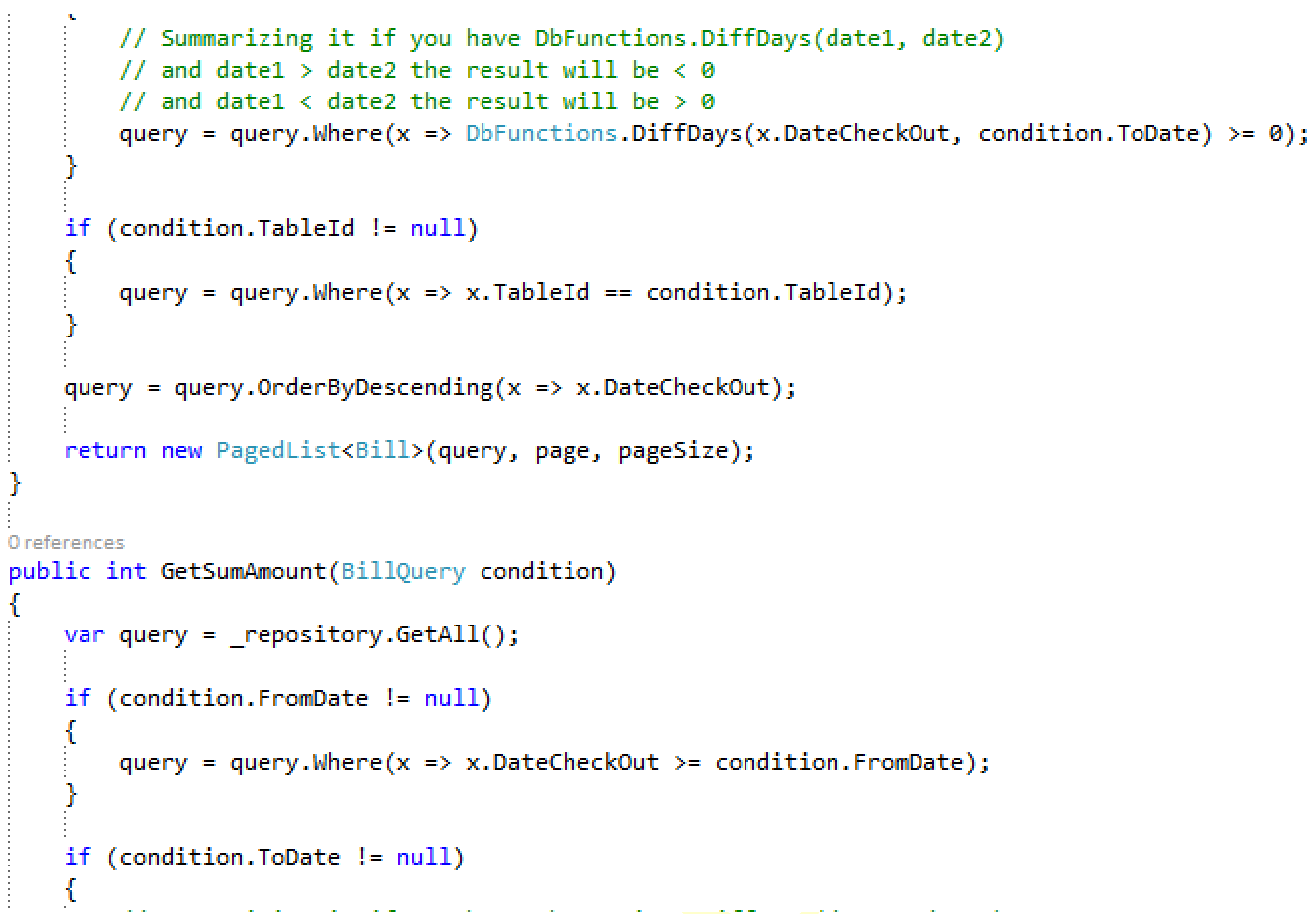
```
public abstract class CrudService<T> : ICrudService<T> where T : class, new()
{
    protected readonly IUnitOfWork _unitOfWork;
    protected readonly IRepository<T> _repository;

    2 references
    public CrudService(IUnitOfWork unitOfWork)
    {
        _unitOfWork = unitOfWork;
        _repository = _unitOfWork.GetRepository<T>();

    }

    4 references
    public virtual void Delete(T entity)
    {
        _repository.Delete(entity);
        _unitOfWork.SaveChanges();
    }

    1 reference
    public virtual T Delete(int id)
    {
        T temp = _repository.Delete(id);
        if(temp != null)
            _unitOfWork.SaveChanges();
        return temp;

    }
    1 reference
    public virtual T GetById(int id)
    {
        return _repository.GetById(id);
    }

    1 reference
    public virtual void Insert(T entity)
    {
        _repository.Insert(entity);
        _unitOfWork.SaveChanges();

    }

    3 references
    public virtual void Update(T entity)
    {
        _repository.Update(entity);
        _unitOfWork.SaveChanges();
    }
```



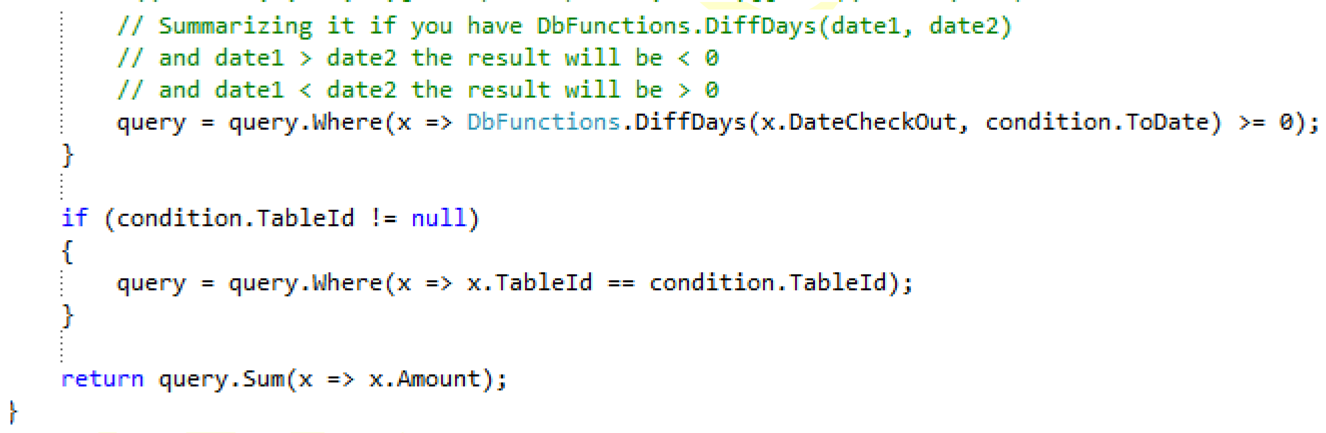
```
5 references
public interface IBillingService : ICrudService<Bill>
{
    2 references
    void CreateOrder(Bill bill);

    3 references
    BillDetail AddBillItem(int billId, Food food, int quantity);

    6 references
    Bill GetCurrentBillForTable(int tableId);

    3 references
    List<BillDetail> GetBillDetails(int billId);

    2 references
    IPagedList<Bill> Search(BillQuery condition, int page = 0, int pageSize = 50);

    1 reference
    bool TransferTable(int sourceTableId, int targetTableId);

    2 references
    bool MergeBill(int sourceTableId, int destTableId);
}
3 references
public class BillingService : CrudService<Bill>, IBillingService
{

    private readonly IRepository<BillDetail> _billDetailRepo;

    1 reference
    public BillingService(IUnitOfWork unitOfWork) : base(unitOfWork)
    {

        _billDetailRepo = _unitOfWork.GetRepository<BillDetail>();
    }

    2 references
    public void CreateOrder(Bill bill)
    {
        _repository.Insert(bill);
        _unitOfWork.SaveChanges();
    }

    3 references
```



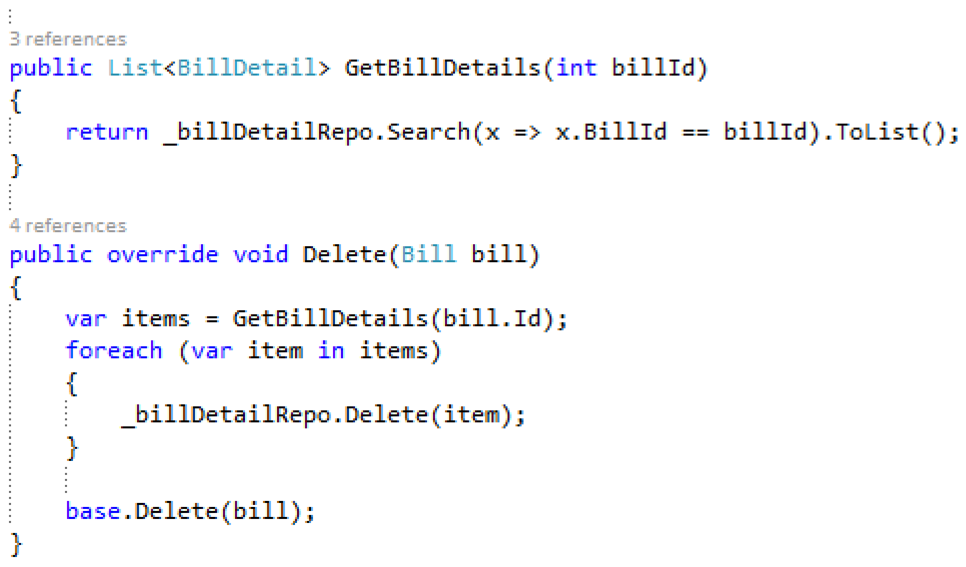
```
public BillDetail AddBillItem(int billId, Food food, int quantity)
{
    var billItem = _billDetailRepo.GetFirst(x => x.BillId == billId && x.FoodId == food.Id);

    if (billItem == null)
    {
        billItem = new BillDetail
        {
            FoodId = food.Id,
            Quantity = quantity,
            UnitPrice = food.UnitPrice,
            BillId = billId
        };

        _billDetailRepo.Insert(billItem);
    }
    else
    {
        billItem.Quantity += quantity;
        _billDetailRepo.Update(billItem);
    }

    _unitOfWork.SaveChanges();

    return billItem;
}

 6 references
 public Bill GetCurrentBillForTable(int tableId)
 {
     var bill = _repository.GetFirst(x => x.TableId == tableId && x.Status == BillStatus.Pending);

     if (bill != null)
     {
         bill.Items = _billDetailRepo.Search(x => x.BillId == bill.Id).ToList();
     }


     return bill;
 }

 2 references
 public IPagedList<Bill> Search(BillQuery condition, int page = 0, int pageSize = 50)
 {
     var query = _repository.GetAll();

     if (condition.FromDate != null)
     {
         query = query.Where(x => x.DateCheckOut >= condition.FromDate);
     }

     if (condition.ToDate != null)
     {
```



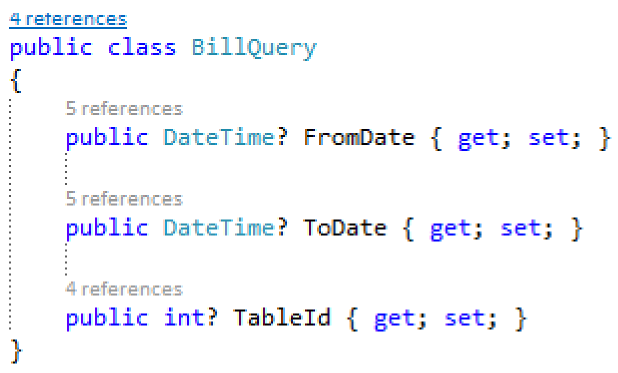
```
        // Summarizing it if you have DbFunctions.DiffDays(date1, date2)
        // and date1 > date2 the result will be < 0
        // and date1 < date2 the result will be > 0
        query = query.Where(x => DbFunctions.DiffDays(x.DateCheckOut, condition.ToDate) >= 0);
    }

    if (condition.TableId != null)
    {
        query = query.Where(x => x.TableId == condition.TableId);
    }

    query = query.OrderByDescending(x => x.DateCheckOut);

    return new PagedList<Bill>(query, page, pageSize);
}

0 references
public int GetSumAmount(BillQuery condition)
{
    var query = _repository.GetAll();

    if (condition.FromDate != null)
    {
        query = query.Where(x => x.DateCheckOut >= condition.FromDate);
    }

    if (condition.ToDate != null)
    {
```



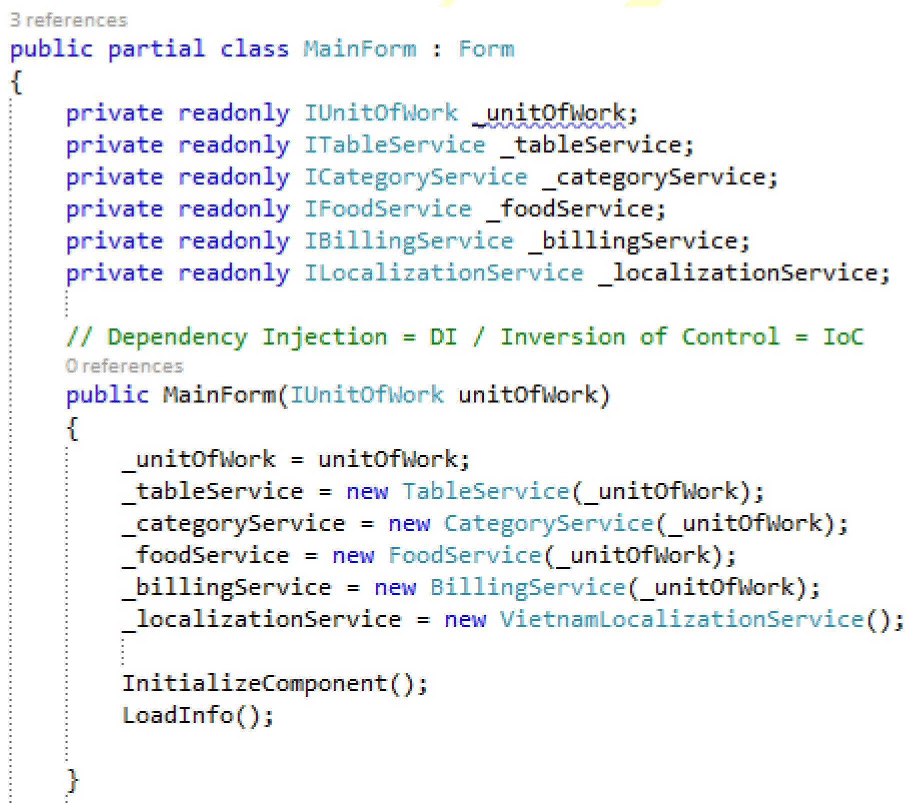
```
        // Summarizing it if you have DbFunctions.DiffDays(date1, date2)
        // and date1 > date2 the result will be < 0
        // and date1 < date2 the result will be > 0
        query = query.Where(x => DbFunctions.DiffDays(x.DateCheckOut, condition.ToDate) >= 0);
    }

    if (condition.TableId != null)
    {
        query = query.Where(x => x.TableId == condition.TableId);
    }

    return query.Sum(x => x.Amount);
}
```



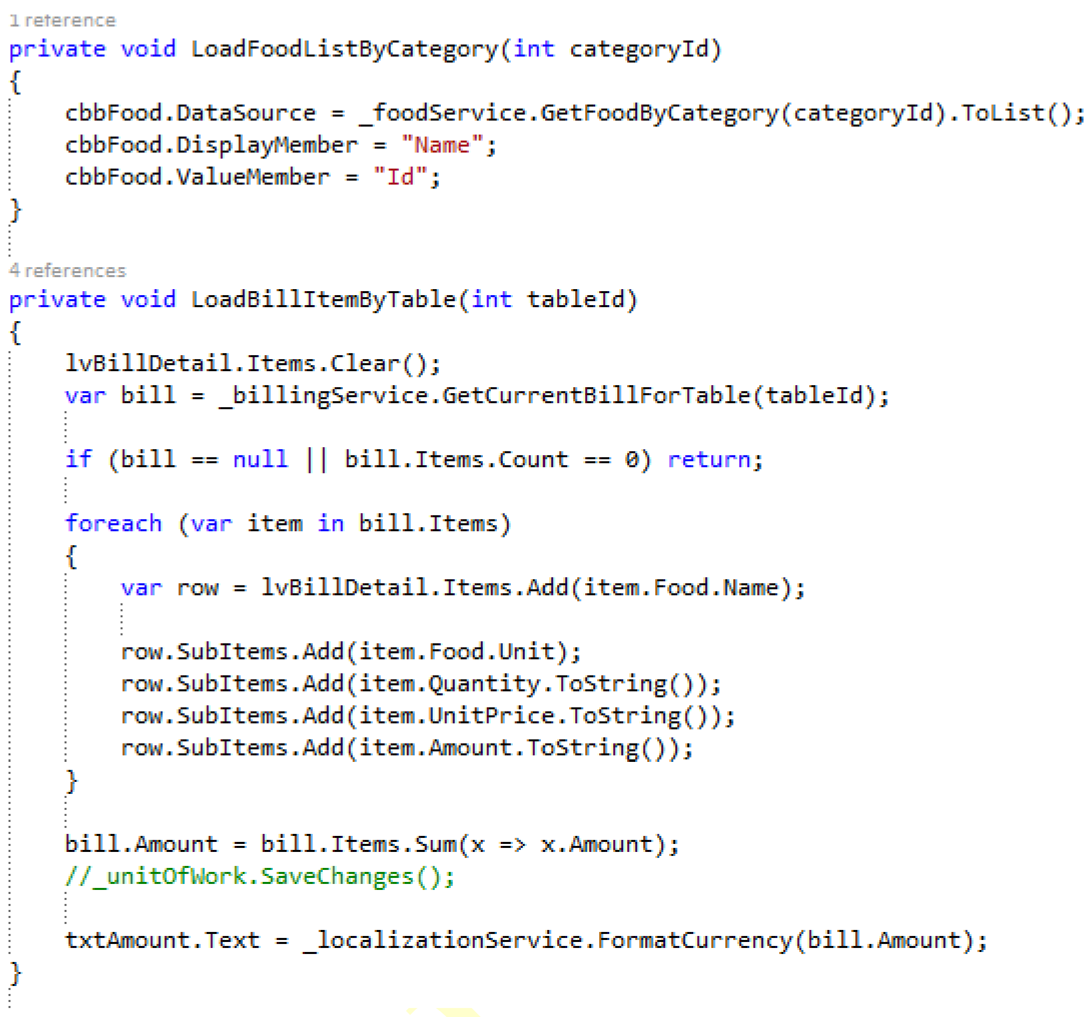
```
3 references
public List<BillDetail> GetBillDetails(int billId)
{
    return _billDetailRepo.Search(x => x.BillId == billId).ToList();
}

4 references
public override void Delete(Bill bill)
{
    var items = GetBillDetails(bill.Id);
    foreach (var item in items)
    {
        _billDetailRepo.Delete(item);
    }

    base.Delete(bill);
}
```



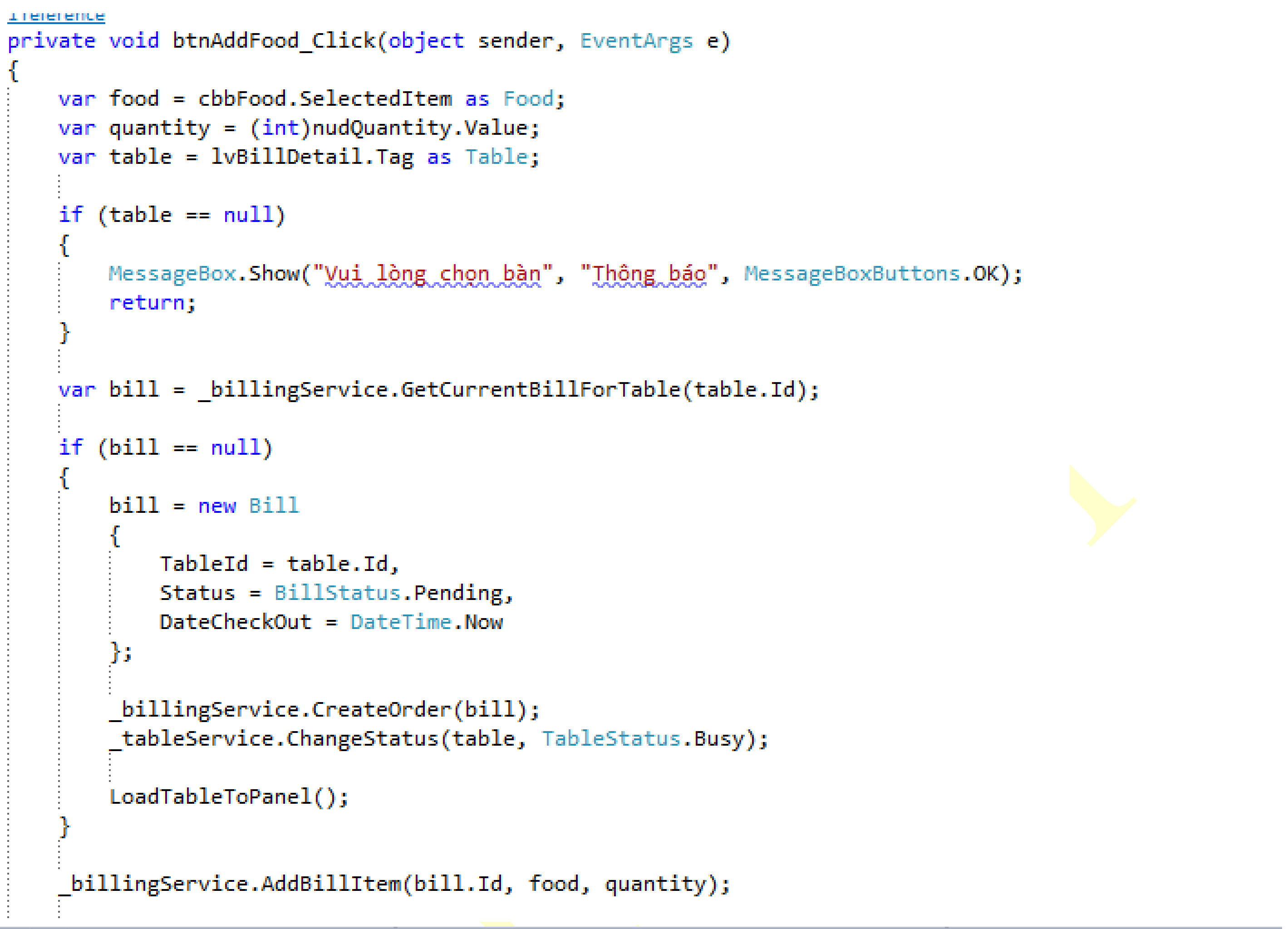
```
4 references
public class BillQuery
{
    5 references
    public DateTime? FromDate { get; set; }

    5 references
    public DateTime? ToDate { get; set; }

    4 references
    public int? TableId { get; set; }
}
```



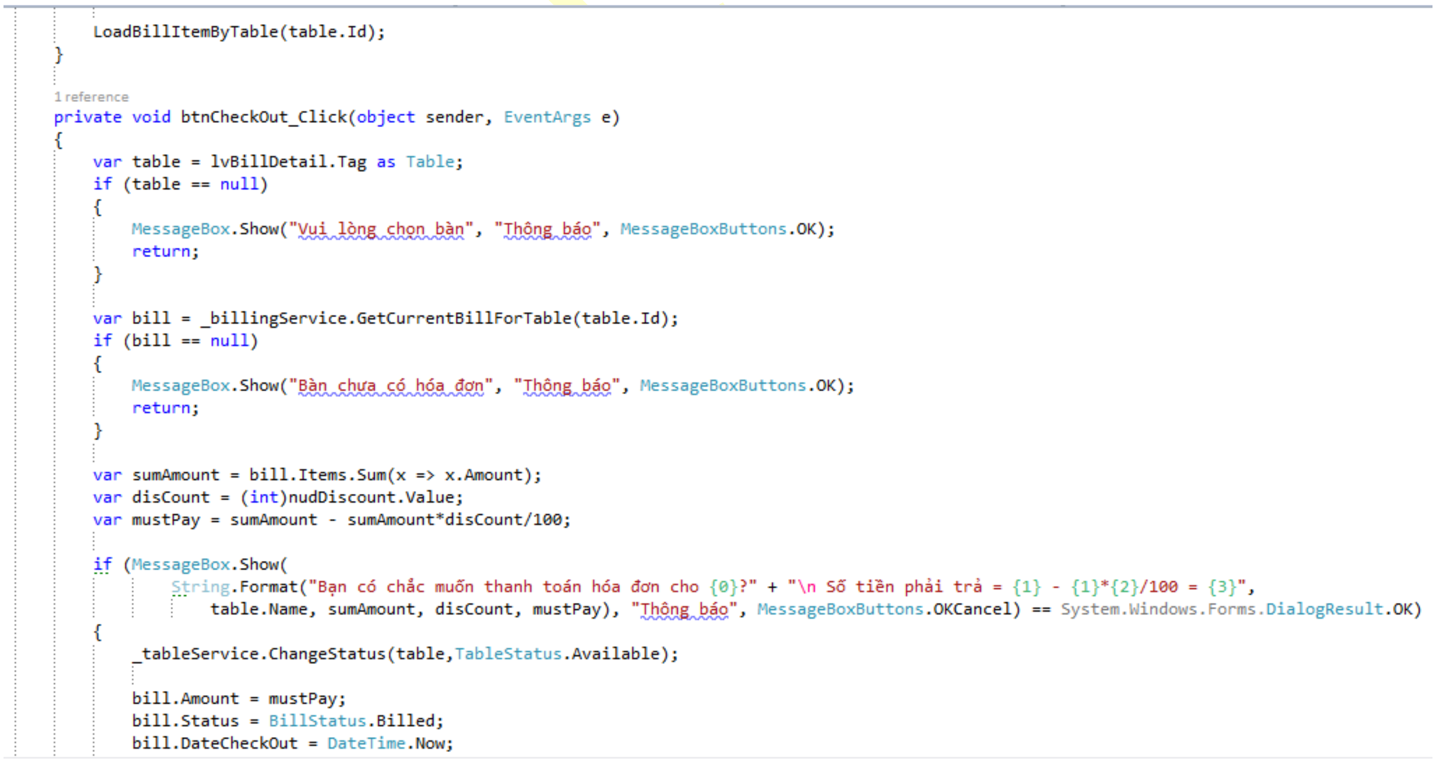
```
3 references
public partial class MainForm : Form
{
    private readonly IUnitOfWork _unitOfWork;
    private readonly ITableService _tableService;
    private readonly ICategoryService _categoryService;
    private readonly IFoodService _foodService;
    private readonly IBillingService _billingService;
    private readonly ILocalizationService _localizationService;

    // Dependency Injection = DI / Inversion of Control = IoC
    0 references
    public MainForm(IUnitOfWork unitOfWork)
    {
        _unitOfWork = unitOfWork;
        _tableService = new TableService(_unitOfWork);
        _categoryService = new CategoryService(_unitOfWork);
        _foodService = new FoodService(_unitOfWork);
        _billingService = new BillingService(_unitOfWork);
        _localizationService = new VietnamLocalizationService();

        InitializeComponent();
        LoadInfo();

    }
```



```
1 reference
private void LoadFoodListByCategory(int categoryId)
{
    cbbFood.DataSource = _foodService.GetFoodByCategory(categoryId).ToList();
    cbbFood.DisplayMember = "Name";
    cbbFood.ValueMember = "Id";
}

4 references
private void LoadBillItemByTable(int tableId)
{
    lvBillDetail.Items.Clear();
    var bill = _billingService.GetCurrentBillForTable(tableId);

    if (bill == null || bill.Items.Count == 0) return;

    foreach (var item in bill.Items)
    {
        var row = lvBillDetail.Items.Add(item.Food.Name);

        row.SubItems.Add(item.Food.Unit);
        row.SubItems.Add(item.Quantity.ToString());
        row.SubItems.Add(item.UnitPrice.ToString());
        row.SubItems.Add(item.Amount.ToString());
    }

    bill.Amount = bill.Items.Sum(x => x.Amount);
    //_unitOfWork.SaveChanges();

    txtAmount.Text = _localizationService.FormatCurrency(bill.Amount);
}
```

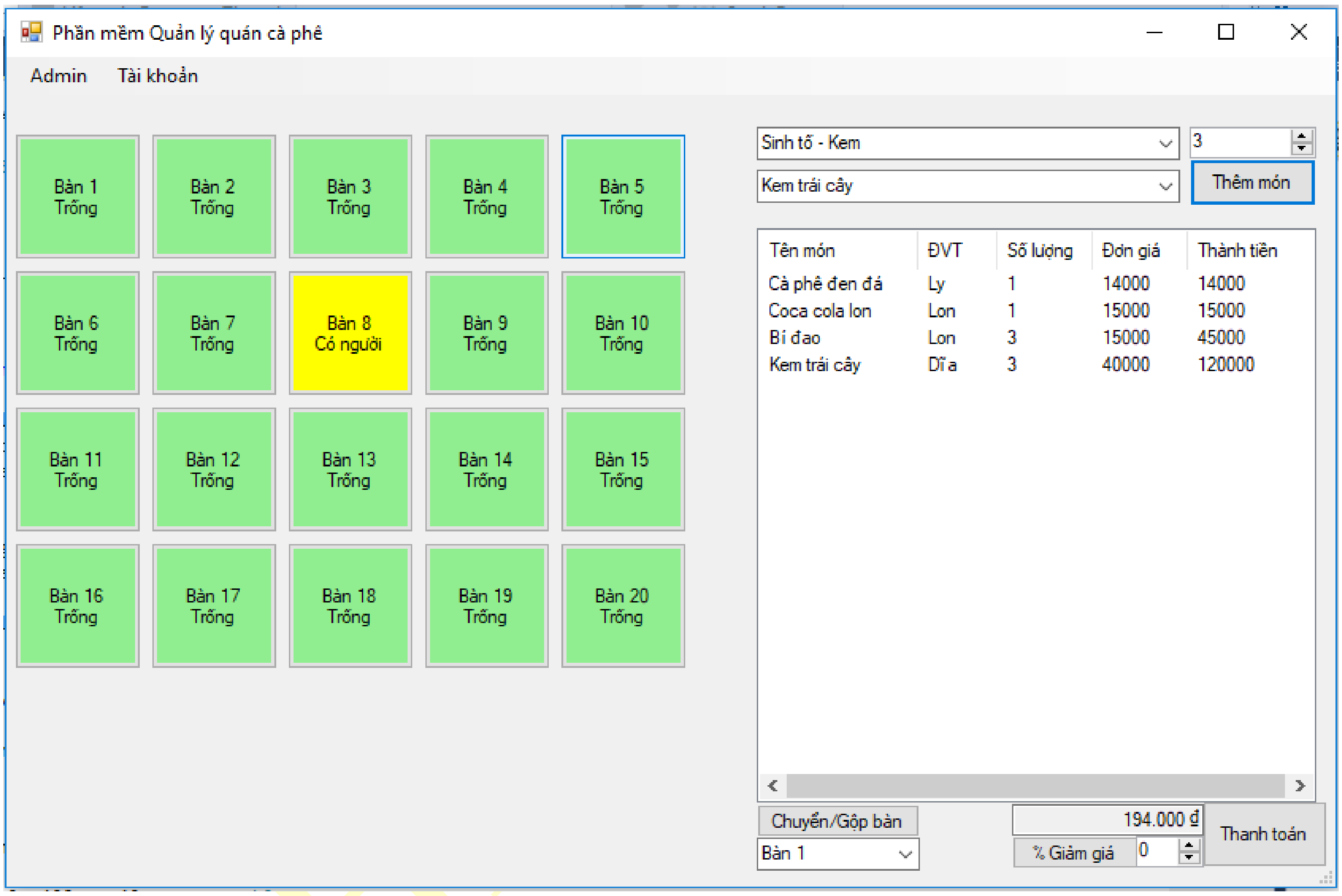


```
1 reference
private void btnAddFood_Click(object sender, EventArgs e)
{
    var food = cbbFood.SelectedItem as Food;
    var quantity = (int)nudQuantity.Value;
    var table = lvBillDetail.Tag as Table;

    if (table == null)
    {
        MessageBox.Show("Vui lòng chọn bàn", "Thông báo", MessageBoxButtons.OK);
        return;
    }

    var bill = _billingService.GetCurrentBillForTable(table.Id);

    if (bill == null)
    {
        bill = new Bill
        {
            TableId = table.Id,
            Status = BillStatus.Pending,
            DateCheckOut = DateTime.Now
        };

        _billingService.CreateOrder(bill);
        _tableService.ChangeStatus(table, TableStatus.Busy);

        LoadTableToPanel();
    }

    _billingService.AddBillItem(bill.Id, food, quantity);
```



```
        LoadBillItemByTable(table.Id);
    }

    1 reference
    private void btnCheckOut_Click(object sender, EventArgs e)
    {
        var table = lvBillDetail.Tag as Table;
        if (table == null)
        {
            MessageBox.Show("Vui lòng chọn bàn", "Thông báo", MessageBoxButtons.OK);
            return;
        }

        var bill = _billingService.GetCurrentBillForTable(table.Id);
        if (bill == null)
        {
            MessageBox.Show("Bàn chưa có hóa đơn", "Thông báo", MessageBoxButtons.OK);
            return;
        }

        var sumAmount = bill.Items.Sum(x => x.Amount);
        var disCount = (int)nudDiscount.Value;
        var mustPay = sumAmount - sumAmount*disCount/100;

        if (MessageBox.Show(
                String.Format("Bạn có chắc muốn thanh toán hóa đơn cho {0}?" + "\n Số tiền phải trả = {1} - {1}*{2}/100 = {3}",
                    table.Name, sumAmount, disCount, mustPay), "Thông báo", MessageBoxButtons.OKCancel) == System.Windows.Forms.DialogResult.OK)
        {
            _tableService.ChangeStatus(table,TableStatus.Available);

            bill.Amount = mustPay;
            bill.Status = BillStatus.Billed;
            bill.DateCheckOut = DateTime.Now;
            bill.Discount = disCount;
            _billingService.Update(bill);

            nudDiscount.ResetText();
            LoadTableToPanel();
            LoadBillItemByTable(table.Id);
        }
    }
```

Giao diện chương trình



Tên món	ĐVT	Số lượng	Đơn giá	Thành tiền
Cà phê đen đá	Ly	1	14000	14000
Coca cola lon	Lon	1	15000	15000
Bí đao	Lon	3	15000	45000
Kem trái cây	Dĩa	3	40000	120000