C# Original - Refactored Code

Original Code

```
if (expense.TransactionType == 1)
{
    cmd.Parameters["@Amount"].Value = -expense.Amount;
}
else
{
    cmd.Parameters["@Amount"].Value = expense.Amount;
}

cmd.Parameters.Add("@Comments", SqlDbType.VarChar, 200);
```

Refactored Code

```
cmd.Parameters["@Comments"].Value = string.IsNullOrEmpty(expense.Comments)
 ? DBNull.Value : expense.Comments;
```

Original Code

```
 if (!string.IsNullOrEmpty(expense.Comments))
 {
     cmd.Parameters["@comments"].Value = expense.Comments;
 }
 else
 {
     cmd.Parameters["@comments"].Value = DBNull.Value;
 }
```

Refactored Code

```
cmd.Parameters["@Comments"].Value = string.IsNullOrEmpty(expense.Comments)
? DBNull.Value : expense.Comments;
```

Original Code

```
  private bool HasScreenPermission(int screenID)
  {
      bool tmp = false;
      if (user.IsSysAdmin) return true;
      foreach (var item in user.Security_Screens)
      {
          if (item.ScreenID == screenID)
          {
              tmp = true;
              break;
          }
          else
          {
              tmp = false;
          }
      }
      return tmp;
  }
```

Refactored Code

```
private bool HasScreenPermission(string screenName)
{
  if (user.IsSysAdmin) return true;
  return user.Security_Screens.Any(item =>
  item.Name.Equals(screenName, StringComparison.CurrentCultureIgnoreCase));
}
```

Original Code

```
private bool HasScreenPermission(string screenName)
{

    if (user.IsSysAdmin) return true;
    bool tmp = false;
    screenName = screenName.ToLower();
    foreach (var item in user.Security_Screens)
    {
        if (item.Name.Equals(screenName, StringComparison.CurrentCultureIgnoreCase))
        {

            tmp = true;
            break;
        }
        else
        {
            tmp = false;
        }
    }
    return tmp;
}
```

Refactored Code

```
private bool HasScreenPermission(string screenName)
{
    if (user.IsSysAdmin) return true;
    return user.Security_Screens.Any(item =>
    item.Name.Equals(screenName, StringComparison.CurrentCultureIgnoreCase));
}
```

Original Code - Primary Constructor

```
   public class DetailsOps
  {
      public int DetailsID { get; set; }
      public int OpID { get; set; }
      public DetailsOps(int _DetailsID, int _OpID)
     {
          DetailsID = _DetailsID;
          OpID = _OpID;
     }
  }
```

Refactored Code - Primary Constructor

```
public class DetailsOps(int _DetailsID, int _OpID)
  {
      public int DetailsID { get; set; } = _DetailsID;
      public int OpID { get; set; } = _OpID;
  }
```

Original Code

```
  public class CWO
    {
        public string Number { get; set; }
        public string Name { get; set; }
        public string WBS { get; set; }

        public string Display
        {
            get
            {
                if (string.IsNullOrEmpty(WBS))
                    return Number + " (" + Name + ") " + "No WBS";
                else
                {
                    return Number + " (" + Name + ") " + WBS;
                }
            }
        }
    }
```

Refactored Code

```
public class CWO
{
    public string Number { get; set; }
    public string Name { get; set; }
    public string WBS { get; set; }
    public string Display =>
    $"{Number} ({Name}) {(string.IsNullOrEmpty(WBS) ? "No WBS" : WBS)}";
}
```

Original Code

```
public bool IsPEAdmin
{
    get
    {
        bool tmp = false;
        foreach (var item in this.Security_Groups)
        {
            if (item.GroupID == 7)
            {
                tmp = true;
            }
            else
            {
                tmp = false;
            }
        }
        return tmp;
    }
}
```

Refactored Code

```
public bool IsPEAdmin => this.Security_Groups.Any(item => item.GroupID == 7);
```

Original Code

```
private static bool HasCellChanged(DataRow row, DataColumn col)
{
    if (!row.HasVersion(DataRowVersion.Original))
    {
        // Row has been added. All columns have changed.
        return true;
    }
    if (!row.HasVersion(DataRowVersion.Current))
    {
        // Row has been removed. No columns have changed.
        return false;
    }
    var originalVersion = row[col, DataRowVersion.Original];
    var currentVersion = row[col, DataRowVersion.Current];
    if (originalVersion == DBNull.Value && currentVersion == DBNull.Value)
    {
        return false;
    }
    else if (originalVersion != DBNull.Value && currentVersion != DBNull.Value)
    {
        return !originalVersion.Equals(currentVersion);
    }
    return true;
}
```

Refactored Code

```
private static bool HasCellChanged(DataRow row, DataColumn col)
{
    if (!row.HasVersion(DataRowVersion.Original))
    {
        // Row has been added. All columns have changed.
        return true;
    }

    if (!row.HasVersion(DataRowVersion.Current))
    {
        // Row has been removed. No columns have changed.
        return false;
    }

    object originalVersion = row[col, DataRowVersion.Original];
    object currentVersion = row[col, DataRowVersion.Current];

    return !Equals(originalVersion, currentVersion);
}
```

Original Code

```
 string UserName = SystemInformation.UserName.ToString().ToUpper();
 switch (UserName)
 {
```

```
     case "PJ":
     case "PAULJ":
     case "ADMINISTRATOR":
     case "ILACQUA":
     case "PAUL":
     case "User":
         break;
 }
```

Refactored Code

```
var adminUsers = new HashSet<string> { "KATHY", "HARRY",
"JOHN", "SMITHY", "PETE", "KEVIN" };
string userName = SystemInformation.UserName.ToString().ToUpper();
 if (adminUsers.Contains(userName))
        {
            userName = "Admin";
        }
```

Original Code

```
 private string _Description;
public string Description
{
get { return _Description; }

set
{
string tmp = _Description == null ? null :
             value.Replace("\n", "").Replace("\r", "").Trim();
_Description = tmp;
}
}
```

Refactored Code

```
 private string _description;
 public string Description
 {
     get => _description;
     set => _description = value?.Replace("\n", "").Replace("\r", "").Trim();
 }
```