User Story:

As a Scrum team member, I want to refactor some code which is really hard to read and with incorrect intent.

Motivation:

Many codebases that severely hurts readability and almost always conveys incorrect intent. This type of code does the most damage because of

the assumptions made on it. Assumptions lead to incorrect maintenance code written, which in turn leads to bugs.

**Before:**

public class Company

{

public void Checkout (IEnumerable<Product> products, Emlpoyee employee)

{

if (! employee.IsNotFlagged)

{

// the customer account is flagged

// log some errors and return

return;

}

if (! employee.IsNotPresent)

{

return;

}

}

}

public class Emlpoyee

{

public decimal Balance {get; private set;}

public bool Present {get; private set;}

public bool IsNotFlagged

{

get {return Balance < 30m;}

}

public bool IsNotPresent

{

get {return Present = true;}

}

}

public class Product

{

}

**Mechanics:**

Here in this example, I am going to remove double negative logic by refactoring.

**After:**

public class Company

{

public void Checkout(IEnumerable<Product> products, Emlpoyee employee)

{

if (employee.IsFlagged)

{

return;

}

if (employee.IsPresent)

{

return;

}

}

}

public class Emlpoyee

{

public decimal Balance {get; private set;}

public bool Present {get; private set;}

public bool IsFlagged

{

get {return Balance < 30m;}

}

public bool IsPresent

{

get {return Present = true; }

}

}

public class Product

{

}