User Story:

As a Scrum team member, I want to refactor a larger switch statement that continually changes because of new conditions being added.

Motivation:

Using a switch statement sometimes implies spaghetti and very crowded code. The Strategy Pattern helps to divide an algorithm from a host class and then move it to another class.

**Before:**

class Program

{

public enum Title

{

Mr,

Mrs,

Doctor,

}

static void Main(string[] args)

{

Title objtitle = Title.Doctor;

switch (title)

{

case Title.Mr:

// do something

break;

case Title.Mrs:

// do something

break;

case Title.Doctor:

// do something

break;

default:

break;

}

}

}

**Mechanics:**

The Strategy Pattern can also help us to replace a switch statement. Strategy Pattern can prevent the horror of using an endless switch or spaghetti code.

**After:**

public enum ePassengerTitle

{

Mr,

Mrs,

Doctor,

}

class Program

{

static void Main(string[] args)

{

ePassengerTitle title = ePassengerTitle.Doctor;

Context.DoAlgorithm(title);

}

}

public interface IPassengerTitleStrategy

{

void DoAlgorithm();

}

public class MrPassengerTitleStrategy : IPassengerTitleStrategy

{

public void DoAlgorithm()

{

throw new NotImplementedException();

}

}

public class MrsPassengerTitleStrategy : IPassengerTitleStrategy

{

public void DoAlgorithm()

{

throw new NotImplementedException();

}

}

public class DoctorPassengerTitleStrategy : IPassengerTitleStrategy

{

public void DoAlgorithm()

{

throw new NotImplementedException();

}

}

public class Context

{

private static Dictionary<ePassengerTitle, IPassengerTitleStrategy> \_strategies =

new Dictionary<ePassengerTitle, IPassengerTitleStrategy>();

public static Context()

{

\_strategies.Add(ePassengerTitle.Mr, new MrPassengerTitleStrategy());

\_strategies.Add(ePassengerTitle.Mrs, new MrsPassengerTitleStrategy());

\_strategies.Add(ePassengerTitle.Doctor, new DoctorPassengerTitleStrategy());

}

public static void DoAlgorithm(ePassengerTitle title)

{

\_strategies[title].DoAlgorithm();

}}