**Replace Type Code with Class**

**Scenario**

The royal Darwin hospital has recently been receiving a large increase in their number of patients. The patient visits to the doctors has increase by 50% as a result, RDH require a system to be able to rapidly search and display the patient blood type. This system will aid the medical field with blood transfusion for patient.

**Reasoning for refactoring**

The integer type is used to indicate the blood type, instead of separating the data type. The problem with this is that the medical system the class cannot contains type code. These values will not affect the behaviour of the system because they value is not used in operator conditions. The solution for this is create a news class and call out it’s object instead of using the type code values. This is also security issues for the system as the internal methods are not private on the blood group but public class.

**Advantages**

Utilises object-oriented programming: sets of primitive values (coded types) into full-fledged classes.

By replacing type code with classes,

allow type hinting for values passed to methods by replacing type code with classes

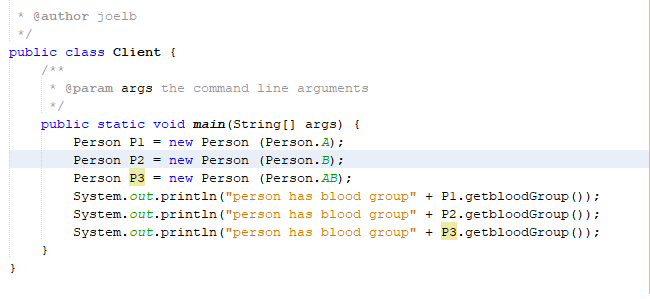
Easy to now move code to the classes of the type in case of the complex manipulation of type value

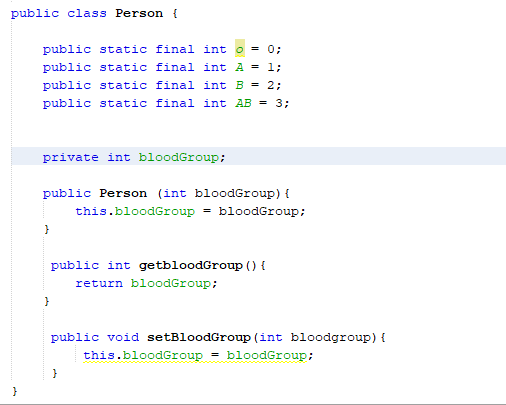
**Disadvantage**

This cannot be done if the code type is inside a control flow structure eg (switch or if statement)

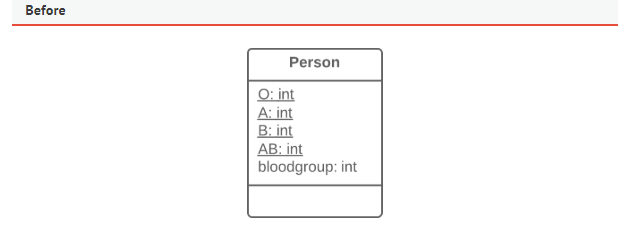
**Current state of code**

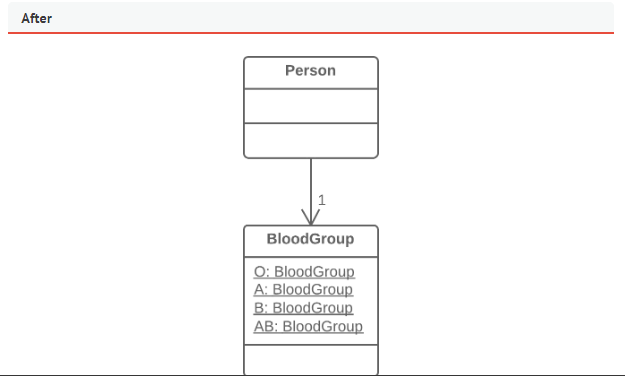
The integer type is used to indicate the blood type





**Before and after state UML**

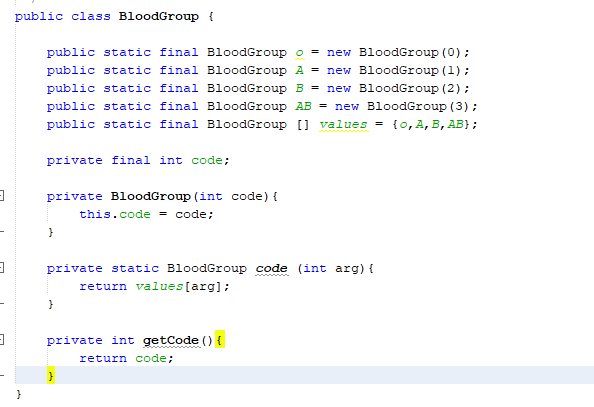




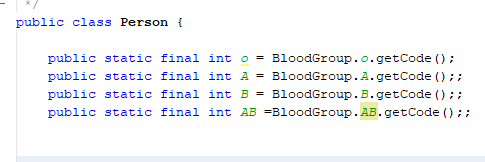
**After (refactored code)**

**Refactoring steps**

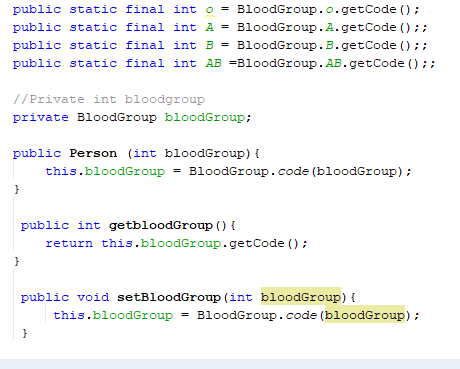
1. Creating the blood group class



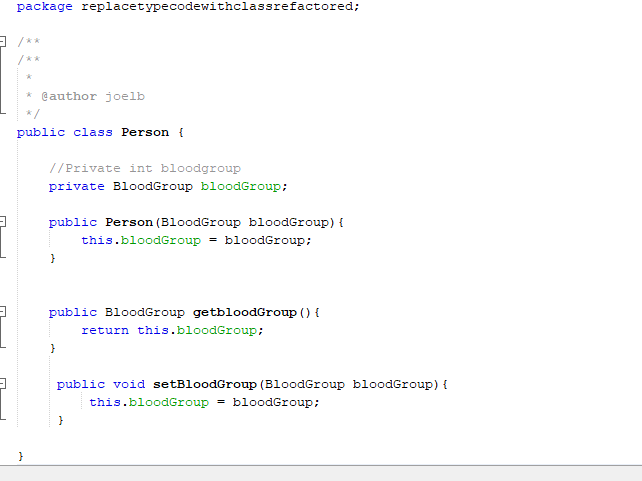
1. Replace the integer by the blood group code



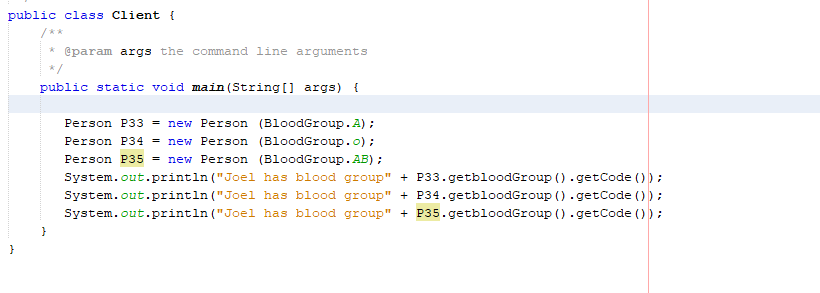
1. Set the blood group value to the input we received
2. Rename the method get blood group to get blood group code
3. Set the blood group type to return the blood code



1. Change the Set blood group method to set the blood code type within the class
2. Create new methods for each value of the coded type to start using the blood group class and remove the old method that call out the integers type.
3. create a new type class object corresponding to this value of the coded type

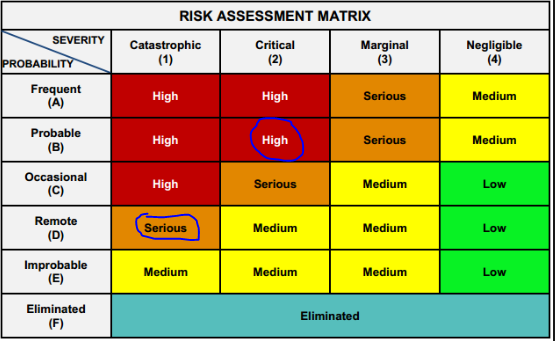


1. Call the methods in the main client class if all is functional



1. We can change all public class to private.
2. We can then add constraint method if necessary

**Risk**

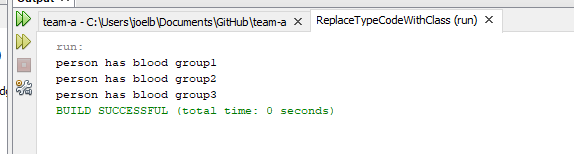
This is the risk assessment matrix to determine the severity of refactoring the program.

High risk because it is a health system

**Testing**

Iterative testing method is important when refactoring a medical system. Refactor small bits at a time and maintaining the system functionality. This ensure that we utilise continuous integration and the application is always ready for use a deployment. However, the are several components that can be tested through unit testing. Component such as the blood type code, out of correct blood group method can be tested.

**Output before refactoring**



**After**

