## Summary

The client requested an application to perform functions related to calculation of tax for different types of employee. The user requirements are listed in the table below, along with an evaluation of how the application meets them.

The application imports data from csv files regarding employee hours and rates, and has a method for determining whether an employee is a resident or working holiday visa holder.

The application writes its calculations to console and exports a csv file using the naming convention specified by the client.

All user requirements were met using the external library CsvHelper.

|  |  |
| --- | --- |
| Specification | Evaluation |
| [Specification as listed in the software specifications] | [How well does your application meet this specification?] |
| 1. The application needs to import a comma delimited (.csv) file for payroll data and calculate the gross, net and tax amounts for each employee. | Application satisfies this requirement. Values are calculated correctly. |
| 1. The comma delimited file contains data for two types of employees:    1. Residents (employees who live in Australia)    2. Working holiday (employees who are visiting Australia and on a working holiday). | Application satisfies this requirement. The application distinguishes types of employee when reading csv file. |
| 1. The method for calculating tax is different for each type of employee; the program must account for this and calculate the tax accurately according to the employee type (see section on calculating tax below). | Application satisfies this requirement. Different calculations are performed based on employee type. |
| 1. Once the pay amounts have been calculated, the employee ID, gross, net and tax amounts for each employee must be written to:    1. a comma delimited file (.csv) using the naming convention {DateTime.Now.Ticks}-records.csv       1. For example: ***637050122993045755-export.csv***       2. For more information on ticks visit the web page [DateTime.Ticks Property](https://docs.microsoft.com/en-us/dotnet/api/system.datetime.ticks?view=netframework-4.8).    2. the console window. | Application satisfies this requirement. Calculations are written to a csv file using the naming convention, and to console. |
| 1. The functionality for writing the computed pay values to the console and a .csv file must be implemented using a **reusable component.** | Application satisfies this requirement. Reusable component CsvHelper is installed. |
| 1. The IT company prides itself on providing high-quality software to customers and requires that unit tests be created for testing by:    1. importing the data from the comma delimited values file and loading the data into a list of objects    2. checking the correctness of the values calculated for the gross, net and tax amounts of each employee. | Application satisfies this requirement. See Testing Documentation. All tests for import, calculations, and export passed. |

Class PayRecord was established as a parent class to ResidentPayRecord and WorkingHolidayPayRecord. For this reason it was made abstract and its children classes inherited from it:



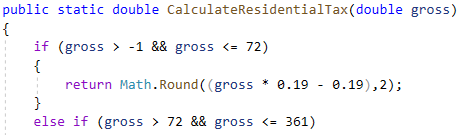




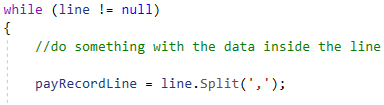
The logic behind this decision was that the children classes inherit:

* the two protected arrays of doubles that store values for hours and rates.
* Properties for calculating Gross and Net values.
* Method GetDetails, which is made virtual and can be overloaded to display additional values in WorkingHolidayPayRecord.

The methods in TaxCalculator class for calculating the value of gross were made static since no object of that class was to be instantiated.



The method ImportPayRecords makes use of .Split:



This is because we are working with a csv file. This line of code reads the commas in each line and splits the values into separate data.