## Evaluation

Evaluation of the product:

My product is evaluated against two groups of criteria laid out in the planning phase.

The first criteria is to "develop a prototype system" that "will receive input of both income and basic tax rules, and output the tax owed or refund to be released" with the functions of

a. *calculate income taxes based on their income*
b. *provide an interface to edit and update tax rules*
c. *store the tax rates set by the individual*
d. *store the income data inputted by the client*
e. *Indicate whether the individual would receive a rebate and specify the value*
f. *be easy to navigate for taxpayers*
g. *display error messages for clients wherever needed*

Functions a and b are achieved by the client stream, c by the tax rules database, d by the income database, and g is done when appropriate. Function f is achieved by the straightforward client dialogue in the client stream. However, e is missing, because a rebate depends on how the tax has been collected, and collected tax has not been taken into consideration at this stage of development.

The second criteria is extensibility for future programmers, so that "the system should be easy to expand and customize. It should lend itself easily to incorporate new tax rules, as the tax rules change almost every year". This is mostly achieved by the modular design of the program (into four packages), the encapsulation of databases and their public interfaces, as well as the very abstract definition of tax rules (including human readable part and backend part, both in strings).

Recommendations for further development:

1. The administrator stream is still under development.
2. The income structure needs to accommodate tax already collected in such a manner as "*IncomeExtended extends income {private int taxCollected…}*. In this way, the missing function e would be implemented.
3. Currently, the three databases involved are stored in arraylist structures. At the next stage, they would be ported over to production databases.
4. For the current stage, a programmer needs to code the tax rule in java for the computer to calculate tax on an income. For the next stage, when SQL databases are applied, the administrator will be able to convert a human-readable tax rule into SQL statements stored in the backend part of the tax rule. The thus-translated SQL statements may be executed against income databases to get a tax result. In an ideal situation, the human-readable tax rule should be translated automatically. However, the semi-automatic manner exemplified by the more realistic SQL statement translation may persist for a long time.

For further development, a system can be implemented to directly transfer the user tax rules into computer-readable without the need for an admin to physically translate the input into computer-readable code.

Words: 453