Victoria University of Wellington
School of Engineering and Computer Science

SWEN221: Software Development
Lab Handout

A Simple Database

The primary purpose of this lab is to get used to working with exceptions. Before the end of the lab, you should submit your solutions via the *online submission system* which will automatically mark it. You may submit as many times as you like in order to improve your mark and the final deadline will be Friday @ 23:59.
In this lab, you will be working with a very simple database implementation. The database reads data from files stored in a particular plain text format, and then allows that data to be accessed from memory. An example database file is:

```
ID:int*,Name:str,Age:int
1,John,88
2,Amy,12
3,Sophie,28
4,John,23
```

The first line of the file describes the database "schema". That is, the layout which the remaining lines must follow. This includes the number of items which must be provided on each line, along with their permitted type (either an `Integer` or a `String`). The asterisk indicates which is the "key field" for the database. This is a special field which uniquely identifies each row such that no two rows may have the same value for the key field, though they can for other fields (e.g. as for the two johns above).

Getting Started

To get started, download the file `database.jar` from course website. This contains the following files:

```
swen221/database/io/DatabaseFileReader.java
swen221/database/lang/ColumnType.java
swen221/database/lang/RowType.java
swen221/database/lang/Database.java
swen221/database/testing/DatabaseTests.java
```

You should import these files into Eclipse. Note, you will find that they do not compile properly yet.

Activity 1 — Adding Exceptions

The first part of the lab is to create the missing classes `InvalidRowException`, `InvalidKeyException` and `DuplicateKeyException` in the package `swen221.database.lang`. Having correctly created these classes, you should find that the code now successfully compiles. However, it will not yet pass any tests.

Activity 2 — Reading Database Files

The second part of the lab is to complete method `DatabaseFileReader.findKeyField()`. To do this, you might find the methods `String.split()` and `String.endsWith()` helpful. Having done this, you may now find the first test is passing (though this depends exactly on how you've implemented it).

Activity 3 — Implementing Database

The third part of the lab is to construct a class `DatabaseImpl` which implements the `Database` interface. You should update method `DatabaseFileReader.read()` to return an appropriate instance of `DatabaseImpl`. You will also need to implement all methods from the `Database` interface correctly. Having done this, you should find that all tests now pass.

*

Submission

Your lab solution should be submitted electronically via the *online submission system*, linked from the course homepage. The required files are:

```
swen221/database/io/DatabaseFileReader.java
swen221/database/lang/ColumnType.java
swen221/database/lang/RowType.java
swen221/database/lang/Database.java
swen221/database/lang/DatabaseImpl.java
```

You must ensure your submission meets the following requirements (which are needed for the automatic marking script):

1. **Your submission is packaged into a jar file, including the source code**. *Note, the jar file does not need to be executable.* See the following Eclipse tutorials for more on this:

   `http://ecs.victoria.ac.nz/Support/TechNoteEclipseTutorials`

2. **The names of all classes, methods and packages remain unchanged**. That is, you may add new classes and/or new methods and you may modify the body of existing methods. However, you may not change the name of any existing class, method or package. *This is to ensure the automatic marking script can test your code.*

3. **All JUnit test files supplied for the assignment remain unchanged.** Specifically, you cannot alter the way in which your code is tested as the marking script relies on this. This does not prohibit you from adding new tests, as you can still create additional JUnit test files. *This is to ensure the automatic marking script can test your code.*

4. **You have removed any debugging code that produces output, or otherwise affects the computation.** *This ensures the output seen by the automatic marking script does not include spurious information.*

Note: Failure to meet these requirements could result in your submission being reject by the submission system and/or zero marks being awarded.