COSC 344 Assignment 3

Group 9

Group Leader: Sonny Lim

Group Members: Robert Young, Amy Corkery, Thomas Scott, Haoyang Gao, Jialin Yang

Part 1: Triggers

First Trigger: MODIFY_HERO_LEVEL

This trigger will calculate a hero's level based on the amount of experience they have accrued. It will react to the creation, update, or deletion of a value from the 'exp' column of the Hero table. A change will be applied to the 'HLEVEL' table, reflecting a change in the hero's level. The level is calculated as $\log_2$ experience.

Here is an example that could fire the trigger:

```
INSERT INTO hero VALUES (101, 'Batman', TO_DATE('02-12-2016','DD-MM-YYYY'), 8888);
UPDATE hero SET exp = 7777 where pid = 101;
DELETE FROM hero WHERE pid = 101;
```

Second Trigger: MODIFY_WEAPONVALUE

This trigger will calculate a weapon's value based on its physical damage, magical damage, and rarity. The trigger is run when there is a creation, update, or deletion applied to the Weapon table's 'mdamage', 'pdamage', or 'rarity' attributes. The weights of the physical and magical damage on the weapon's value are dependent on its rarity attribute.

Here is an example that could fire the trigger:

```
INSERT INTO weapon VALUES (101, ' Sword1', 2, 0, 'Common', 1, 'BarMoneyPack', 'Save the Princess');
UPDATE weapon SET mdamage = 111 WHERE wid = 101;
DELETE FROM weapon WHERE wid = 101;
```

Third Trigger: MODIFY_MONSTERHP

This trigger will calculate a monster's hit points based on its level. The trigger runs when there is an addition, update, or deletion applied to the 'MHP' table, reflecting a change in the monster's hitpoints. A monster's hitpoints are calculated as 1.2 * level + 100.

Here is an example that could fire the trigger:

INSERT INTO monster VALUES(‘Hello kitty’, 100,’Castle of Doom’);
UPDATE monster SET mlevel = 200 WHERE mname=’Hello Kitty’;
DELETE FROM monster WHERE mname = ‘Hello Kitty’;

Fourth Trigger: CHECK_HERO_EXP

This trigger will run before a insertion or update is applied to the ‘exp’ column of the ‘HERO’ table. If this value is negative, an exception is thrown.

Here is an example that could fire the trigger:
INSERT INTO hero VALUES (101, 'Batman', TO_DATE('02-12-2016','DD-MM-YYYY'), 8888);
UPDATE hero set exp = -1 where pid = 101;

Fifth Trigger: CHECK_MON_LEVEL

This trigger will run before a insertion or update is applied to the ‘mlevel’ column of the ‘MONSTER’ table. If this value is negative, an exception is thrown.

Here is an example that could fire the trigger:

INSERT INTO monster VALUES('Hello kitty',100,'Castle of Doom');
UPDATE monster SET mlevel = -1 WHERE mname='Hello Kitty';

Part 2: Program

Our database is based on the idea of storing data for the Diablo video game, so the idea behind the program is a ‘command line version’ of Diablo. We have implemented it using Java.

This project is well-architectured with the following 3 packages:

- Models: POJO file as model classes, including all the necessary database entities (Area, Backpack, Consumable, Hero, Monster, Quest and Weapon).
- Services: The DAO wrapper around the models above. They provide the CRUD operations from the database in separate classes. They also support the search feature or something like groupBy (MonsterService)
- Utilities: Responsible for various things needed to run the game, such as setting up the connection and generating ascii images of the game elements, a single class to handle the game logic.

How to operate:

- Data Preparation
    - Execute load.sql in the database.
    - Update the file named password.txt in the /utils/ folder with the following content:
        - Line 1: Database link (jdbc:oracle:thin:@silver:1527:cosc344)
        - Line 2 : User name
        - Line 3 : password
- Execution:
    - Cd src
    - Javac cosc344/Main.java
    - Java cosc344.Main
    - Follow the on-screen prompts.

A sample of input that is useful in operating the program:

- Step 1: Choose a hero by the id :
    - 1
- Step 2: Choose a weapon by the id:
    - 2
- Step 3: Choose an area by its name:
    - Undead land
- Step 4: battle start:
    - p : physical damage;
    - m : magic damage;

        - q : quit;
- Step 5: If you quit or all monsters are dead, show the report.

Part 3: Teamwork Summary

All group members contributed equally. The report was a mixed, collaborative effort.

Amy was responsible for the idea of having the program be command-line Diablo, and conceived some of the base code for that project. Albert constructed the rest of it, and finalised the Java code in the package.

Thomas was responsible for researching how to get SQL working with Java, including the JDBC library, the database connection (UserPass and so on), and some information on methods of executing a SQL statement and processing the result.

Jialin and Robert took responsibility for devising what the triggers would do, and writing them.

Sonny was responsible for organising the group and making sure that we were all present, knew what we were working on and commented on the codes.