Software Engineering Group Project 15

Maintenance Manual

Authors: Samuel Sherar,
Date: 10/02/2013
Version: 1.0
Status: Draft

Table of Contents

Project Description

This program is an online game to play for children and adolescents can play, where they own a “Monster Farm”, and are able to Fight, Breed and Sell these monsters to other players on the server. It is all written in Java, utilising certain modules of J2EE, such as JPA (Java Persistence Architecture) and Glassfish server, and the website is majorly written in xHTML Transitional, CSS2 and Javascript.

Program Structure

The program was developed with a strong MVC. Therefore:

- All data models of the database, including JPA references, should be placed in the model package
- All Servlets should be placed in the view package. Please note that JSPs don’t live in this file, and only reside in the WEBCONTENT directory.
- All controllers, including Data Access Objects and miscalleneous functions should be placed here.

Method calling

The start of a method call always starts from a Servlet. For example, all data sent from the login form gets sent to the LoginServlet. From there, it will check all values using a DAO against the related model. If everything goes well it, the user is sent to the target page, otherwise an error will display to the user.

For more details on the overall design, please view the Design Document.

Algorithms

There are three main algorithms which are important to the game: Aging, Fighting and Breeding.

Aging

For aging the monster, we have designed the solution to work out the age from base values, which are found in the database; from there it will age day by day. We also have age ranges for all monsters: child (0-4), adult(5-11) and elderly(12+); during these phases of the monsters life, different stats will increase or decrease depending on how old it is. The values are as follows:

- Children have a 5% increase to all stats (not including health)
- Adults have a 2% increase to Strength and Aggression; Fertility and Health stay the same
- The elderly monsters stats stay the same, but the health of the monster decreases at 15 - (strength + aggression + fertility7) points each day, until it hits 0

Fighting

The main fighting algorithm has three parts to it: Attack, Health and Luck, which all combine to give a normalised value between which monster will win. The formula for calculating the value is:
hit =Strength Aggression1000 + health20 + rand(1, 10). This is classed as a 1 hit system for the monsters, and the winning monster will have his health decreased by a percentage based on the hit which was given by the other monster.

Breeding

For breeding monsters, we decided that we will follow a simplified version of reproduction in the real world. Instead of having dominant and recessive genes which might pop out at a certain point, we created that there will be an equal chance that child will get the weaker parents stat to getting the stronger parents stat. We also gave an element of randomness for mutations, which will choose and random number between 20 and 70 and not at all biased on the parents statistics

For more details on the overall design, please view the Design Document.

Data Structures

All data structures in this solution (apart from Friend.java) are based on MySQL table entities - and vice versa, so using JPA mapping between the three tables are incredibly easy, and as the structure is written in Java, moving the table structure between databases/servers is simple

For more details on the overall design, please view the Design Document.

Issues and Improvements

All issues and improvements to the software can be found at the Github issue tracker. This is where all maintainers and developers will post up bug issues and new enhancements, which will then be handled by individual maintainers.

Making Changes

When making changes to the codebase, make sure you have forked the repository to your own account on Github, this will make merging requests easier when merging your additional code to the master branch of the main repo. Also, when you are fixing an issue, please assign yourself to the issue!

Physical Limitations

There are several physical limitations of the software at present. One of which is the database server MySQL. If the site does take a large amount of users in a short amount of time, MySQL is not the most efficient server for such a task.

However using JPA shouldn't be a problem with upgrading the database to a NoSQL alternative.

Rebuilding and Testing

To rebuild, I suggest using a J2EE environment which allows for automatic compiling and deploying of the .war file to the glassfish server, such as Netbeans or Eclipse with certain addons. The server requires that you have installed the J-Connector for MySQL into the lib folder of the glassfish server, and that you've created a pool of connections and a driver called 'jdbc/mysql' which has access to the database. (NB: the name of the driver can change if you edit the persistance.xml file).