**CLASS DESCRIPTIONS**

**Monster** – This class contains the attributes of the monster and methods to set monsters for breeding or for sale.

**User** – This class contains the details of the user, including username and password required to log them in. It also contains variables for the amount of money they have and their list of friends. It contains methods to set the password, get money (when account is created, by winning fights, selling, buying and breeding monsters), adding friends and deleting friends, adding monsters (by buying or breeding) and deleting monsters(when they lose fights or die of old age).

**AccountManager** – The class manages the user accounts. It contains methods to create an account, remove an account, reset an account and authenticate the user.

**DataManager** – This class contains methods to obtain data from the server. It stores the value of the player’s ID. The main methods in this class are:

DataManager(String playerID) – This method takes the player’s ID as the parameter, so that it can check whether the user is allowed to send or receive data.

getFriends() – This method returns the player’s list of friends.

getMonsters() – This method returns the player’s list of monsters.

getFriendsMonsters(MonsterFilter f) – This method returns the player’s friends’ monsters and filters it by type (that is, whether it is available for breeding, selling or all monsters in their farm).

getMonster(String id) – This method returns the selected monster.

**BreedResolver** – This class maintains breeding between monsters.

**FightResolver** – This class maintains fights between monsters.

**PurchaseResolver** – This class maintains the sales of monsters between users.

**RequestDispatcher** – This class maintains requests. It takes requests from JSP pages and processes them.

**Request** – It is an abstract class representing general requests.

**BreedRequest, BuyRequest, FightRequest** - These classes represent specific requests.