Project
Lutémon

Team (Finno-Ugric Dream Team):

Eeli Remes, Eeli.Remes@student.lut.fi
Bátor Lang, Bator.Lang@student.lut.fi
Márton Magyar, Marton.Magyar@student.lut.fi

Description:

This project work “Lutemon” is an Android game where users can create Lutemon creatures that battle against AI enemies. Users can train and/or fight with their lutemons to make them stronger. Lutemons get stronger with level. Level increases with experience points (XP) and Lutemons gain stats on level up. The game can be played infinitely since enemies get stronger alongside the user.

We have implemented all the base features for this project but changed some features using our own judgment.

The battle is implemented differently than in the project description. The user does not select two lutemons to battle but chooses one to fight against an enemy whose strength is scaled according to the player’s average Lutemon level. There are appropriate methods to make sure level scaling is fair and can be adjusted for more balanced gameplay. The Lutemons will never die in battle, instead after they are defeated, they are healed to full health and can be used in battle again if user so desires. The reason for making an Enemy class is because we wanted the game to be infinitely scalable, and we could also include “bosses” or high-level special enemies later in development. Special enemies are not currently implemented.

Lutemon training is handled in the training view, and after pressing the train button there is a short delay before the Lutemon will gain XP that makes it stronger. The training method has a cooldown to prevent too frequent training.

Additionally, “moving” the Lutemons is implemented differently in our project, as training is fast and done in a single click and because of the unique battle implementation.

The level-up mechanic is one of our unique features. In the project description XP would increase stats, but in our project, *level* increases stats and makes Lutemons stronger. For example, the Black Lutemon type gains 5 attack, 1 defense and 4 health per level. XP gained from battles depends on the enemy level

Lutemons are stored in a HashMap with the id as key. The HashMap is stored in the Storage class, which is used largely throughout the program. Storage implementation follows the Singleton pattern with eager initialization of the Storage instance.

Our project contains multiple views.

The home view handles lutemon creation and file handling.

When the battle button is pressed, the user is moved to the Lutemon selection view, where user chooses the Lutemon to enter to battle with. In the battle screen the top part has the enemy's HP bar, and the bottom has the user's Lutemon HP. The middle part displays the battle log, which describes damage taken and results.

The training view has a drop-down menu where the user chooses the Lutemon to train. It is also decorated with training-themed imagery.

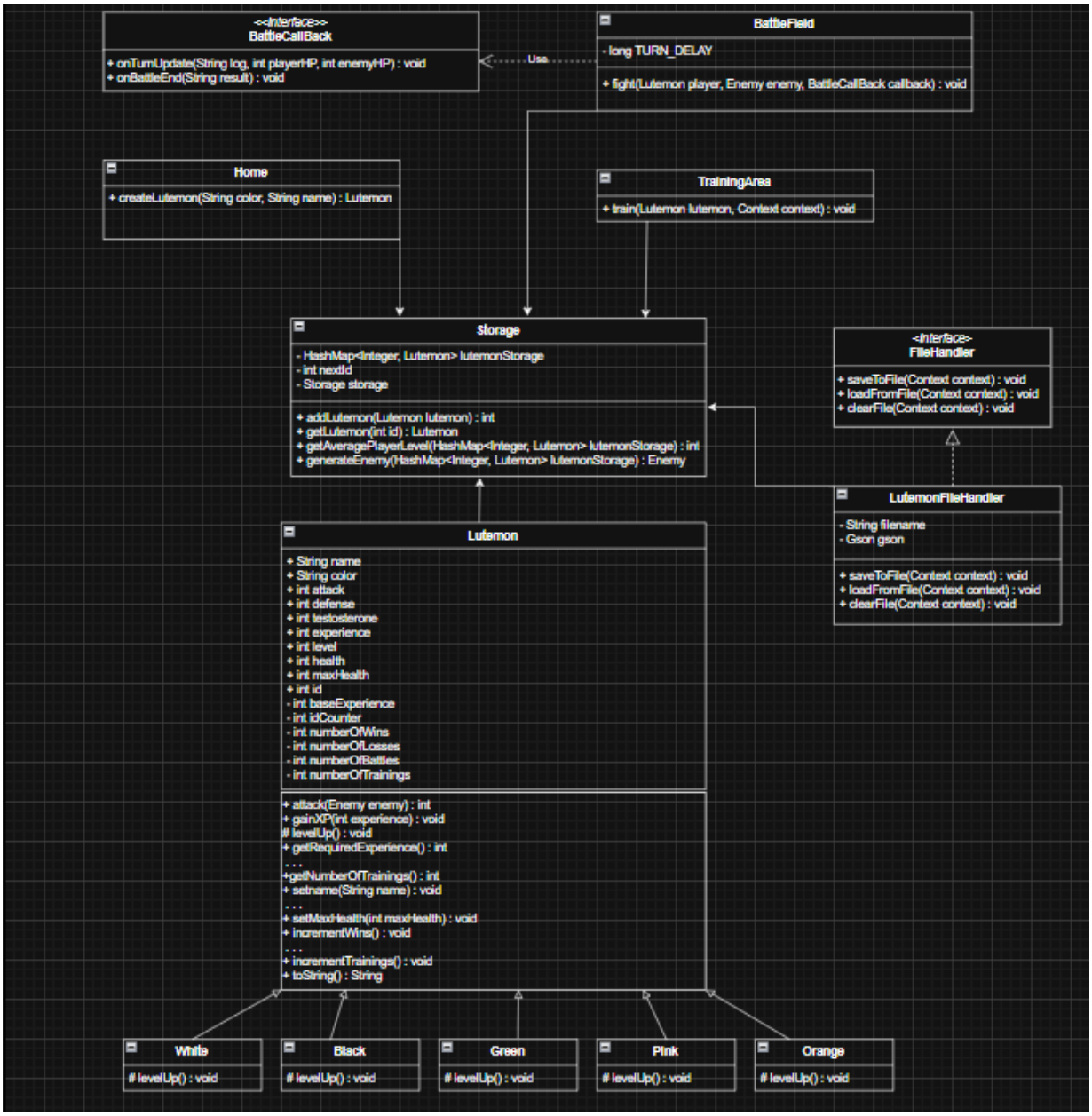
The statistics view is like the Lutemon storage view, but instead of stats, it displays the battles fought, won, lost and times trained for each Lutemon.

The storage view uses a RecyclerView that displays the Lutemons and their stats.

List of Bonus features that we implemented for extra points:

- RecyclerView, is used to list Lutemon details and statistics
- Battle Visualization, Players and enemies have HP bars that visualize their current health.
- Lutemons have images, Lutemon types have unique images representing them. They also have unique training images
- Statistics, Lutemons have statistics for battles won, battles lost, times trained etc.
- No death, defeated Lutemons are healed to full health, and can be used again in battle
- Randomness in Battle, Critical hits (Double damage) are implemented in battle (5% for enemies, Lutemons have a testosterone stat that increases critical chance.) Additionally, Enemy difficulty scaling has slight randomness, and first turn is decided by a coin flip.
- Data Storage & loading, user can load and save their lutemons. User can also clear the file if they so desire. There is only one save file users can work with.
- Custom Feature X, The testosterone stat that is used for critical strike chance for player-owned lutemons. The battle implementation which includes the generation of an Enemy according to player's average level. Additionally, the level scaling instead of XP scaling. These are our "custom" features.

The UML Class Diagram for the project, excluding activities and utility classes for RecyclerView etc. Lutemon class getters and setters are not all listed for convenience. Also, class diagram does not include the Chromatic type Lutemon, which is not used in the current version. Finally, some methods that were left unused are not listed in the UML Diagram.


<<Interface>>
BattleCallBack
+ onTurnUpdate(String log, int playerHP, int enemyHP) : void
+ onBattleEnd(String result) : void
Use
BattleField
- long TURN_DELAY
+ fight(Lutemon player, Enemy enemy, BattleCallBack callback) : void
Home
+ createLutemon(String color, String name) : Lutemon
TrainingArea
+ train(Lutemon lutemon, Context context) : void
Storage
- HashMap<Integer, Lutemon> lutemonStorage
- int nextId
- Storage storage
+ addLutemon(Lutemon lutemon) : int
+ getLutemon(int id) : Lutemon
+ getAveragePlayerLevel(HashMap<Integer, Lutemon> lutemonStorage) : int
+ generateEnemy(HashMap<Integer, Lutemon> lutemonStorage) : Enemy
<<Interface>>
FileHandler
+ saveToFile(Context context) : void
+ loadFromFile(Context context) : void
+ clearFile(Context context) : void
LutemonFileHandler
- String filename
- Gson gson
+ saveToFile(Context context) : void
+ loadFromFile(Context context) : void
+ clearFile(Context context) : void
Lutemon
+ String name
+ String color
+ int attack
+ int defense
+ int testosterone
+ int experience
+ int level
+ int health
+ int maxHealth
+ int id
- int baseExperience
- int idCounter
- int numberOfWins
- int numberOfLosses
- int numberOfBattles
- int numberOfTrainings
+ attack(Enemy enemy) : int
+ gainXP(int experience) : void
# levelUp() : void
+ getRequiredExperience() : int
- - -
+getNumberOfTrainings() : int
+ setname(String name) : void
- - -
+ setMaxHealth(int maxHealth) : void
+ incrementWins() : void
- - -
+ incrementTrainings() : void
+ toString() : String
White
levelUp() : void
Black
levelUp() : void
Green
levelUp() : void
Pink
levelUp() : void
Orange
levelUp() : void