Object-Oriented Programming- Project Work: LUTEMON

The purpose of this project is to build an application that mimics Pokémon – or something similar. In this case, the program features five different types of “Lutemon” (white, green, pink, orange, and black). The user can create and train them, as well as send them to an arena to battle.

The application will be developed as an Android app, where the user can input all necessary values through a graphical user interface and command the Lutemons directly from menus.

NOTE: The instructions given in this document are not the only way to implement the Lutemon game. You can do something similar and use your judgment on how to do the actual implementation.

The project is to be completed in groups of 1-3 students.

There are two submissions in the project work: 1) Project plan and 2) Final project submission.

Project plan

The project plan is a short but concise plan on what you are implementing (the topic), what features it will contain, and who is on your team. The plan should include a UML diagram of your program and a preliminary user interface description.

The deadline for submitting the project plan is March 30, 2025, at 23:59 (Finnish time).

Final project submission

Return a link to the project Github, along with a video describing the program, the project documentation and your team composition.

The deadline for the final project submission is April 20, 2025, at 23:59 (Finnish time).

1 Instructions

The program allows the creation of Lutemons, which are initially placed in “home.” The user can then move them to the training area or the battle arena.

When a Lutemon is trained, it gains experience points that affect its combat skills. For example, if a Lutemon has 2 experience points, its attack power increases by 2 points.

The battle system is turn-based, where one Lutemon attacks another. The attacked Lutemon defends and then retaliates. The battle continues until one Lutemon’s health value drops to zero or below. The winner is the surviving Lutemon, which earns an experience point. The defeated Lutemon dies and is removed from the program.

1.1 Battle Algorithm

A possible algorithm for handling battles could be as follows:

1. Allow the user to select two Lutemons from those available in the battle arena.
2. Enter a **while loop** to execute the battle.
3. Print the stats of both Lutemons (referred to as A and B).
4. Execute the attack operation: B.defense(A); (or alternatively B.defense(A.attack())).
5. If B is still alive:
    a. Print the message **“B managed to avoid death.”**
    b. Swap the roles of A and B (so B attacks next).
6. Otherwise:
    a. Print the message **“B has died.”**
    b. Award an experience point to A.
    c. Return Lutemon A to the battle arena.
    d. Exit the loop.

Once a Lutemon is returned to home, it fully regenerates its health but retains its accumulated experience points.

Figure 1 provides an example class diagram, which can be freely modified and expanded. Table 1 presents example values for different Lutemons.

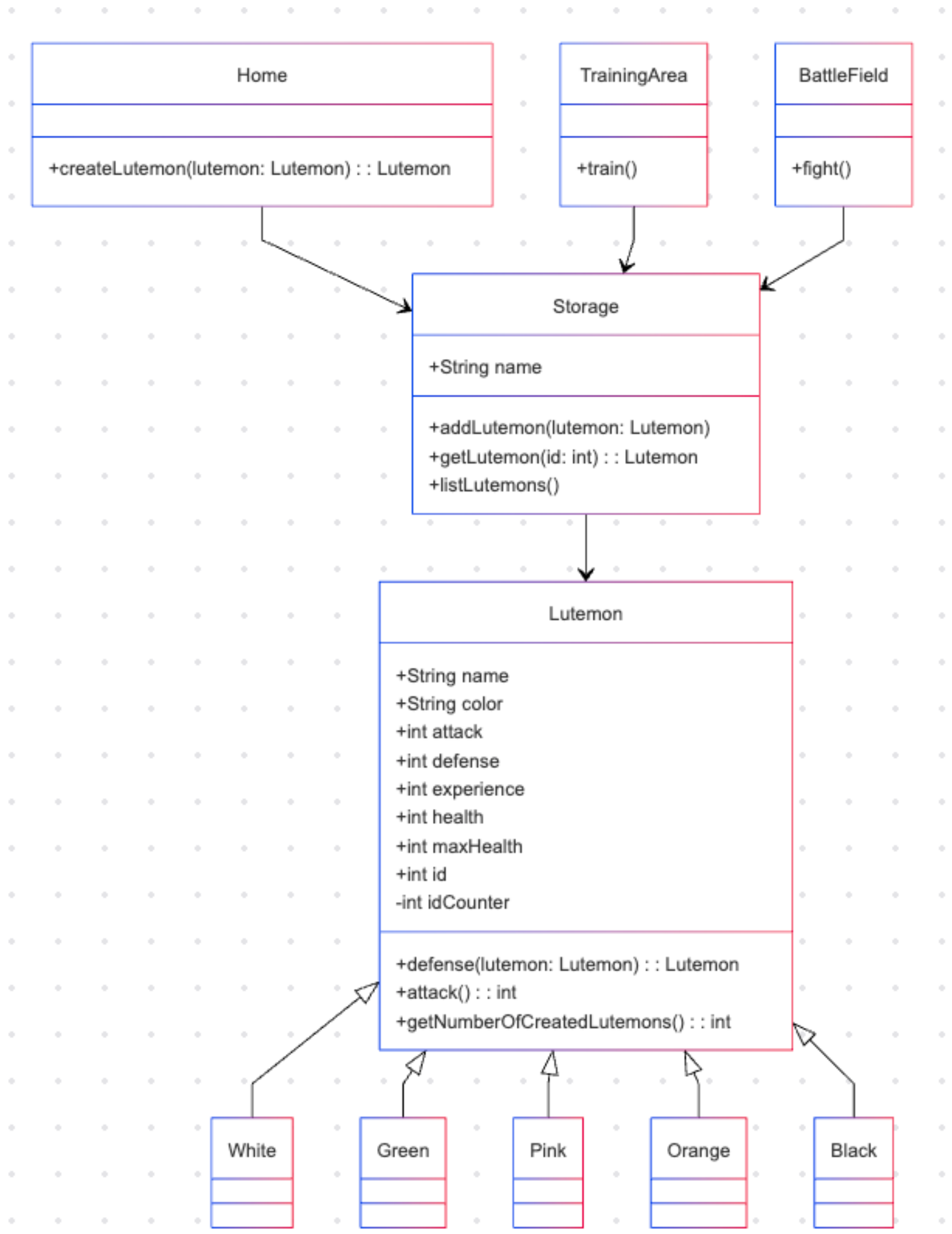
Table 1: Default Values for Lutemons Upon Creation

Lutemon Color	Attack	Defense	Max Health Points
White	5	4	20
Green	6	3	19
Pink	7	2	18
Orange	8	1	17
Black	9	0	16

Experience starts at zero, and health points begin at their maximum.

1.2 Class Diagram (example)

The class diagram could be something like the one shown here. However, it can differ depending on your own implementation.



1.3 Data structures

You can choose which data structures to use, but in the Storage class, a HashMap can be quite handy since it can store both the Lutemon and its ID:

```
HashMap<Integer, Lutemon>
```

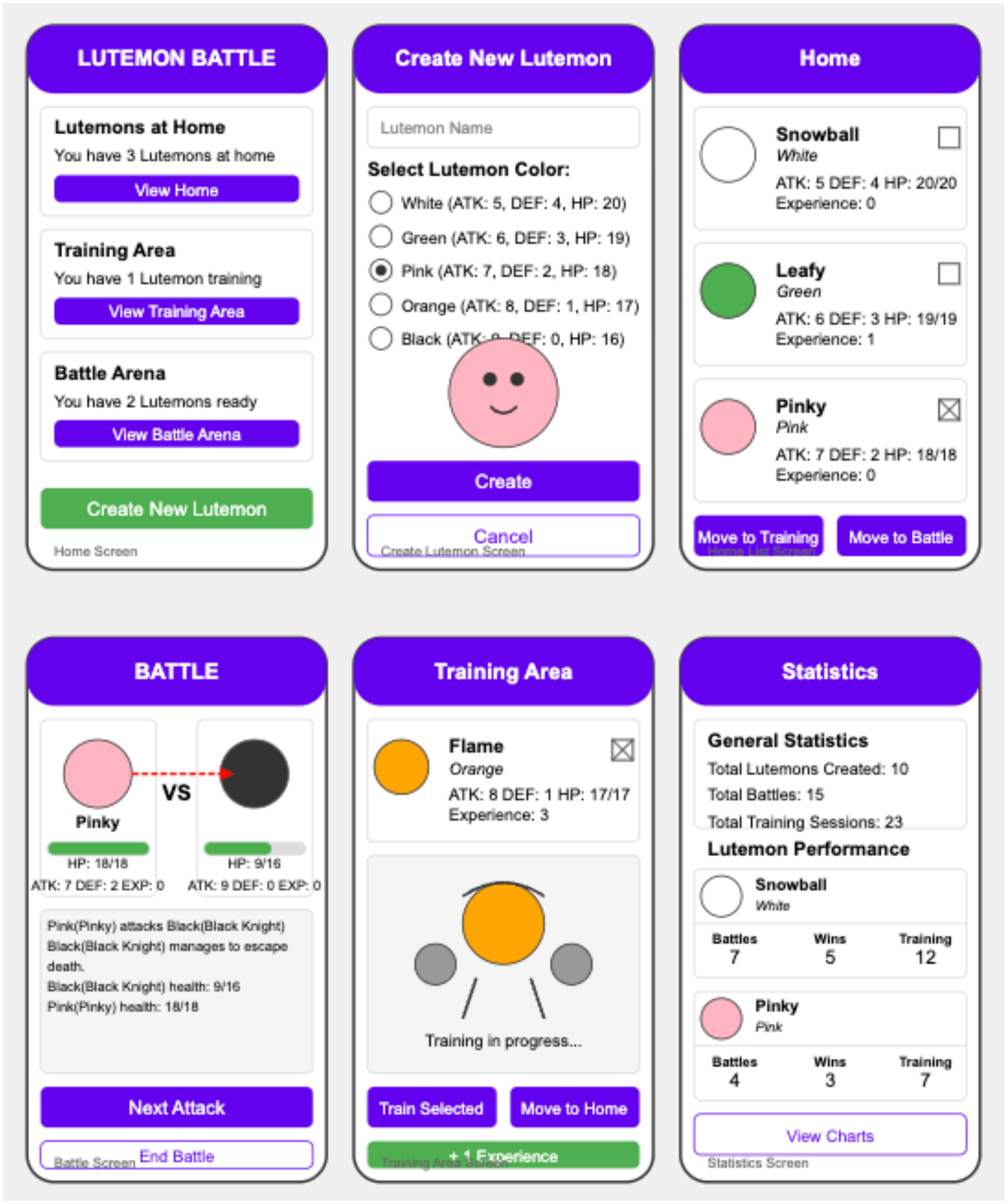
With an ArrayList, you may need to implement various loops to manage the data, but handling it in RecyclerView is more straightforward.

1.4 Example of battle between Lutemons

```
1: Pink(Pinky) att: 7; def: 2; exp:0; health: 18/18
2: Black(Black Knight) att: 9; def: 0; exp:0; health: 16/16
Pink(Pinky) attacks Black(Black Knight)
Black(Black Knight) manages to escape death.
2: Black(Black Knight) att: 9; def: 0; exp:0; health: 9/16
1: Pink(Pinky) att: 7; def: 2; exp:0; health: 18/18
Black(Black Knight) attacks Pink(Pinky)
Pink(Pinky) manages to escape death.
1: Pink(Pinky) att: 7; def: 2; exp:0; health: 11/18
2: Black(Black Knight) att: 9; def: 0; exp:0; health: 9/16
Pink(Pinky) attacks Black(Black Knight)
Black(Black Knight) manages to escape death.
2: Black(Black Knight) att: 9; def: 0; exp:0; health: 2/16
1: Pink(Pinky) att: 7; def: 2; exp:0; health: 11/18
Black(Black Knight) attacks Pink(Pinky)
Pink(Pinky) manages to escape death.
1: Pink(Pinky) att: 7; def: 2; exp:0; health: 4/18
2: Black(Black Knight) att: 9; def: 0; exp:0; health: 2/16
Pink(Pinky) attacks Black(Black Knight)
Black(Black Knight) gets killed.
The battle is over.
```

1.5 UI mockups

Here are some mockup ideas for the user interfaces. Feel free to implement them however you like. This is just to give some ideas to get started with.



LUTEMON BATTLE

Lutemons at Home
You have 3 Lutemons at home
View Home

Training Area
You have 1 Lutemon training
View Training Area

Battle Arena
You have 2 Lutemons ready
View Battle Arena

Create New Lutemon

Home Screen

Create New Lutemon

Lutemon Name

Select Lutemon Color:
White (ATK: 5, DEF: 4, HP: 20)
Green (ATK: 6, DEF: 3, HP: 19)
Pink (ATK: 7, DEF: 2, HP: 18)
Orange (ATK: 8, DEF: 1, HP: 17)
Black (ATK: 9, DEF: 0, HP: 16)

Create
Cancel

Create Lutemon Screen

Home

Snowball
White
ATK: 5 DEF: 4 HP: 20/20
Experience: 0

Leafy
Green
ATK: 6 DEF: 3 HP: 19/19
Experience: 1

Pinky
Pink
ATK: 7 DEF: 2 HP: 18/18
Experience: 0

Move to Training
Move to Battle

Home List Screen

BATTLE

Pinky
VS
HP: 18/18
ATK: 7 DEF: 2 EXP: 0
HP: 9/16
ATK: 9 DEF: 0 EXP: 0

Pink(Pinky) attacks Black(Black Knight)
Black(Black Knight) manages to escape death.
Black(Black Knight) health: 9/16
Pink(Pinky) health: 18/18

Next Attack
End Battle

Battle Screen

Training Area

Flame
Orange
ATK: 8 DEF: 1 HP: 17/17
Experience: 3

Training in progress...

Train Selected
Move to Home
+1 Experience

Training Area Screen

Statistics

General Statistics
Total Lutemons Created: 10
Total Battles: 15
Total Training Sessions: 23

Lutemon Performance

Snowball
White

Battles	Wins	Training
7	5	12

Pinky
Pink

Battles	Wins	Training
4	3	7

View Charts

Statistics Screen

2. Basic requirements

To successfully complete this project and meet the **mandatory requirements**, the following aspects must be implemented:

1. Object-Oriented Programming (OOP) Principles
    - a. The program must be coded according to the object-oriented paradigm.
    - b. Use appropriate **classes** (e.g., Lutemon, Storage, Battle).
    - c. Follow encapsulation, inheritance, and polymorphism where necessary.
2. Code Language
    - d. All code, comments, and documentation must be in **English**.
3. Android App Development
    - e. The application **must run on Android** devices.
    - f. It must be developed using **Java** in **Android Studio**.

b. Basic Functionality. The application should include the following features:
   - a. Lutemon Management
       - i. Users can **create** different types of Lutemons (white, green, pink, orange, black).
       - ii. Newly created Lutemons are placed in **home**.
       - iii. Users can move Lutemons to **training areas** or **battle arenas**.
   - b. Training System
       - i. Lutemons **gain experience points** when trained.
       - ii. Experience increases attack power (e.g., if XP = 2, attack increases by 2 points).
   - c. Turn-Based Battle System
       - i. Users can select **two Lutemons** to battle.
       - ii. Battles follow a **turn-based** system:
           1. One Lutemon **attacks**, the other **defends**.
           2. The battle continues until one Lutemon’s **health drops to zero**.
           3. The **winner gains** an experience point.
           4. The **loser is removed** from the program.
       - iii. A **battle algorithm** is outlined for implementation.
   - d. Lutemon Recovery
       - i. When a Lutemon returns **home**, its **health is fully restored** but experience points remain.

4. Data Structures
    - e. The program must use **data structures** effectively.
    - f. **HashMap**<Integer, Lutemon> could be used for storing Lutemons and their IDs.
    - g. **ArrayList** may be useful for managing lists of Lutemons, especially with RecyclerView.

3 Grading Criteria

Implementing the mandatory features earns 13 points, and additional points can be gained by implementing extra features.

3.1 Mandatory Requirements (Required for 13 Points)

Requirement	Description	Points
Object-Oriented Code	The program is coded according to the object-oriented paradigm.	Required
Code in English	All code and comments must be in English. However, documentation and UI elements can be in Finnish.	Required
Android Compatibility	The program must work on various Android phones and be coded in Java using Android Studio.	Required
Basic Functionality	The program must implement all the features defined above so that users can interact with Lutemons.	Required
Documentation	The final submission must include documentation (PDF or GitHub .md file) containing: class diagram (excluding UI classes like Activities), division of work among team members, implemented features, and a general project description.	Required

3.2 Bonus Features (Extra Points)

Feature	Description	Points
RecyclerView	Uses the RecyclerView component to list Lutemons and their details.	+1
Lutemons Have Images	Different Lutemons are visualized with unique images.	+1
Battle Visualization	By default, a textual description of the battle is displayed in a TextView. However, you can improve this by dynamically adding visual battle effects, such as an attacking Lutemon's image displaying a sword pointing at the defender. Creativity is encouraged!	+2
Statistics	Track how each Lutemon has performed (number of battles, victories, training days, etc.).	+1
No Death	Instead of dying, a defeated Lutemon returns to its initial state. Lost battles can also be tracked in statistics.	+1
Randomness in Battles	Add some randomness to battles, e.g., attack + Math.random() * 3.	+1
Fragments	Use fragments meaningfully in the application.	+2
Data Storage & Loading	Save all Lutemons to a file and allow the user to load them when needed.	+2
Statistics Visualization	Visualize game statistics in some way (e.g., using AnyChart-Android).	+2
Custom Feature X	Implement and grade your own feature!	+0 - 2

3.3 Penalty for Inappropriate Content

Issue	Description	Penalty
Inappropriate Content	If the program, documentation, comments, etc., contain inappropriate material (e.g., trash, hate speech, racism).	-5 to -100

By implementing all required features, the base score is **13 points**. Additional features allow for more points, potentially increasing the final score significantly.

4. Make your project more meaningful?

Does this project feel uninspiring? Would you prefer to create a **useful** application for yourself?

Maybe your **parents** need an app for something?

If you have an idea for a **different project**, suggest it to the lecturer—let's see if it can be turned into your coursework!