LW: Pokemon Trainer

Overview

For years, the Bureau of Pokemon Affairs in the Alola region of the Pokemon world has been tracking the region's registered Pokemon trainers by hand, due to the region's lack of gyms for supporting a large number of citizens in the profession.

However, due to the Alola government's recent construction of several new gyms in the region to train homegrown talent, there has been a strong growth of Pokemon trainers in the region. As a result, the Bureau has hired your team of software developers to plan the initial design of their proposed Pokemon trainer database system.

For the labwork, you will be creating a Unified Modeling Language (UML) diagram for a given set of requirements of a Pokemon database system.

- Refer to **Instructions** for the set of requirements and the design steps to create the UML diagram.
- Refer to **Grading** for details of satisfying labwork completion. Work with others in your group on the labwork, but submit individually on Canvas.
- Refer to **Appendix** for supplementary information on UML concepts.

Objectives

- Convert text requirements into classes, attributes, and methods.
- Design a complete UML diagram based on text requirements.
- Correctly apply UML diagrammatic symbols.

Instructions

Setup

1. Form up into groups of 3-5 people.
2. Have one member of your group create a Google Slides document, and share this document with the remaining members of your group.
3. Create two slides in the Google Slides document.
    a. The first slide will contain the list of names of your group members.
    b. The second slide will contain your UML diagram.

Requirement #1: The Types and Visibility

- The data will be represented with the following types:
- Text: The text representation.
- Number: The numerical representation.
- Boolean: The boolean representation (i.e., true, false).
- The data's visibility will be defined as follows:
    - attributes: These will be private.
    - methods: These will be public.

Requirement #2: The Database

- The database can add a trainer.
- The database can remove a trainer by the trainer's identification number.
- The database's size (i.e., the number of registered trainers) can be accessed.

Requirement #3: The Trainer

- The trainer has a name, identification number, home gym, number of wins, a starter pokemon, and a collection of pokemon.
- The trainer's name, home gym, identification number, starter pokemon, and collection of pokemon can be accessed.
- The trainer's home gym can be modified.
- The trainer's number of wins can be incremented.

Requirement #4: The Gym

- The gym has a name and a gym leader.
- The gym leader is a trainer.
- The gym's name and leader can be accessed.
- The gym's leader can be modified.

Requirement #5: The Pokemon

- The pokemon has a name, an identification number, a type, a level, and whether it has fainted.
- The pokemon's name, identification number, type, level, and fainted state can be accessed.

Requirement #6: The Pokemon Type

- The pokemon type has a name and which pokemon types that it is strong and weak against.
- The pokemon's name and which pokemon type that it is strong and weak against can be accessed.

Requirement #7: The Pokemon Collection

- The number of pokemon in the pokemon collection can be accessed.
- A pokemon can be accessed from the pokemon collection by its identification number.
- Contents from the pokemon collection can be removed from the pokemon itself (i.e. the Pokémon class) and from the pokemon's identification number.

Requirement #8: The Relationships

- A trainer database is an **aggregation** of trainers.
- A trainer has an affiliation with a gym.
- A gym hosts a trainer.
- A trainer owns a starter pokemon.
- A trainer possesses a pokemon collection.
- A pokemon collection is an **aggregation** of pokemon.
- A pokemon is **composed** of its type.

Grading

Submission Grade

You must submit the completed labwork on Canvas as either:

- a valid PDF file, or
- two image files: one image for the list of names, one image for the diagram

Attendance Grade

- Your TA will mark your attendance at the start of class, and will confirm your attendance after your group shows your completed work to your TA.
- If you do not attend lab at the start of class or if you do not receive confirmation of satisfactory completion from your TA when your group submits, then you will not receive lab credit for this week.

Makeup Work

Before you can do any make-up work, you must provide your instructor with any documentation for your excused absence.

Appendix

Concepts related to UMLs were covered in last week's lecture. This section highlights the relevant UML concepts that are used in this labwork. **The Teaching Assistants will overview this section at the start of your lab.**

Classes

- A class is a type of data that stores three types of information: the class name, the attributes, and the methods. It is analogous to classes in C++.
    - Google Slides: You can use a table with three rows and one column.

Attributes

- An attribute is a data field. It is analogous to member variables or data variables in C++ classes.
- Attributes have the following syntax:
    - `attributeName : attributeType`

Methods

- A method is a behavior that can accept input and can return output. It is analogous to functions in C++.
- Methods with no return type have the following syntax:
    - `methodName(attributeName : attributeType, [...])`
- Methods with a return type have the following syntax:
    - `methodName(attributeName : attributeType, [...]) : attributeType`

Visibility

- Publicly-visible data is prepended with a `'+'` sign.
- Privately-visible data is prepended with a `'-'` sign.

Relationships

- Class diagrams can have various relationships with each other. These include: association, aggregation, and composition.
- Association is when one class is connected to or associated with another class.
    - Let's say that **A** eats **B**. Then the arrow will also have a label called "eats".
    - Example: The Panda class eats the Bamboo class.
    - Google Slide: Let's say that a class, **A**, is connected to or associated with another class, **B**. Use a plain arrow that points from **A** to **B**.
- Aggregation is when one class owns or accesses another class.
    - Example: The Zoo class owns the Panda class.
    - Google Slide: Let's say that a class, **A**, owns or accesses another class, **B**. Use an arrow with a clear diamond head that points from **B** to **A**.
- Composition is when one class is composed specifically from another class
    - Example: The Panda class is composed of the Head, Arms, Legs, and Torso classes.
    - Google Slide: Let's say that a class, **A**, is composed of another class, **B**. Use an arrow with a solid diamond head that points from **B** to **A**.