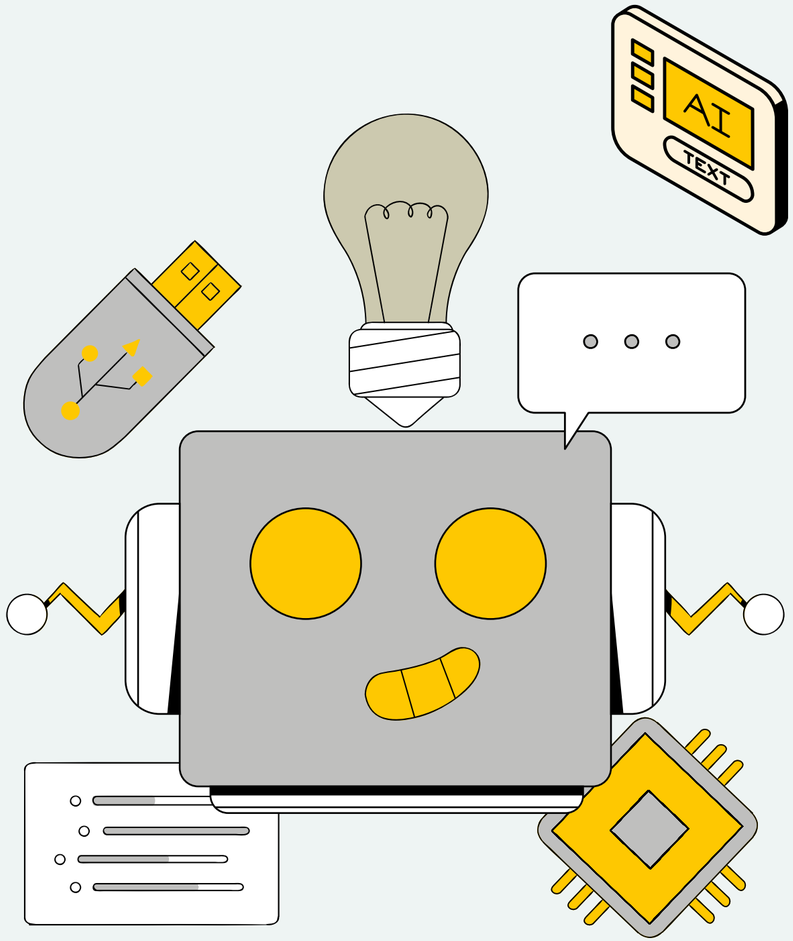


OOP FINAL PROJECT
PRESENTATION

B123245025 李君潔
B123245026 黃昱憙
B123040060 陳泉龍

PRESENTATION OUTLINE

1. **Code Review**
2. **Live/Recorded Demo**
3. **Project Explanation**

   OOP concepts, UML diagrams

CODE REVIEW

LIVE/RECORDED DEMO

PROJECT EXPLANATION

PART 2: EXPLANATION

EnhancedQLearningAgent (Worker)

- which encapsulates the core RL logic, the Q-table state, and the decision-making policies.

EnhancedAgentTrainer (Manager)

- which acts as an orchestrator. It manages the lifecycle of multiple agents, handles the hyperparameter injection, and performs the validation steps.

PART 2: EXPLANATION

Optimistic Initialization (7.0): Encourages early exploration of unvisited states.

UCB Exploration Bonus: Replaces simple epsilon-greedy with count-based curiosity.

The Polish Phase:

- Phase 1 (12k eps): High exploration to map the grid.
- Phase 2 (3k eps): Low exploration (epsilon=0.02) to stabilize the optimal path.

Reward Shaping: Potential-based shaping (Manhattan distance) to guide the agent.

PART 2: OOP CONCEPTS

Abstraction:

- train() and polish() methods hide complex loops, decay math, and UCB calculations.
- Helper methods like _calculate_moving_average() hide statistical operations.

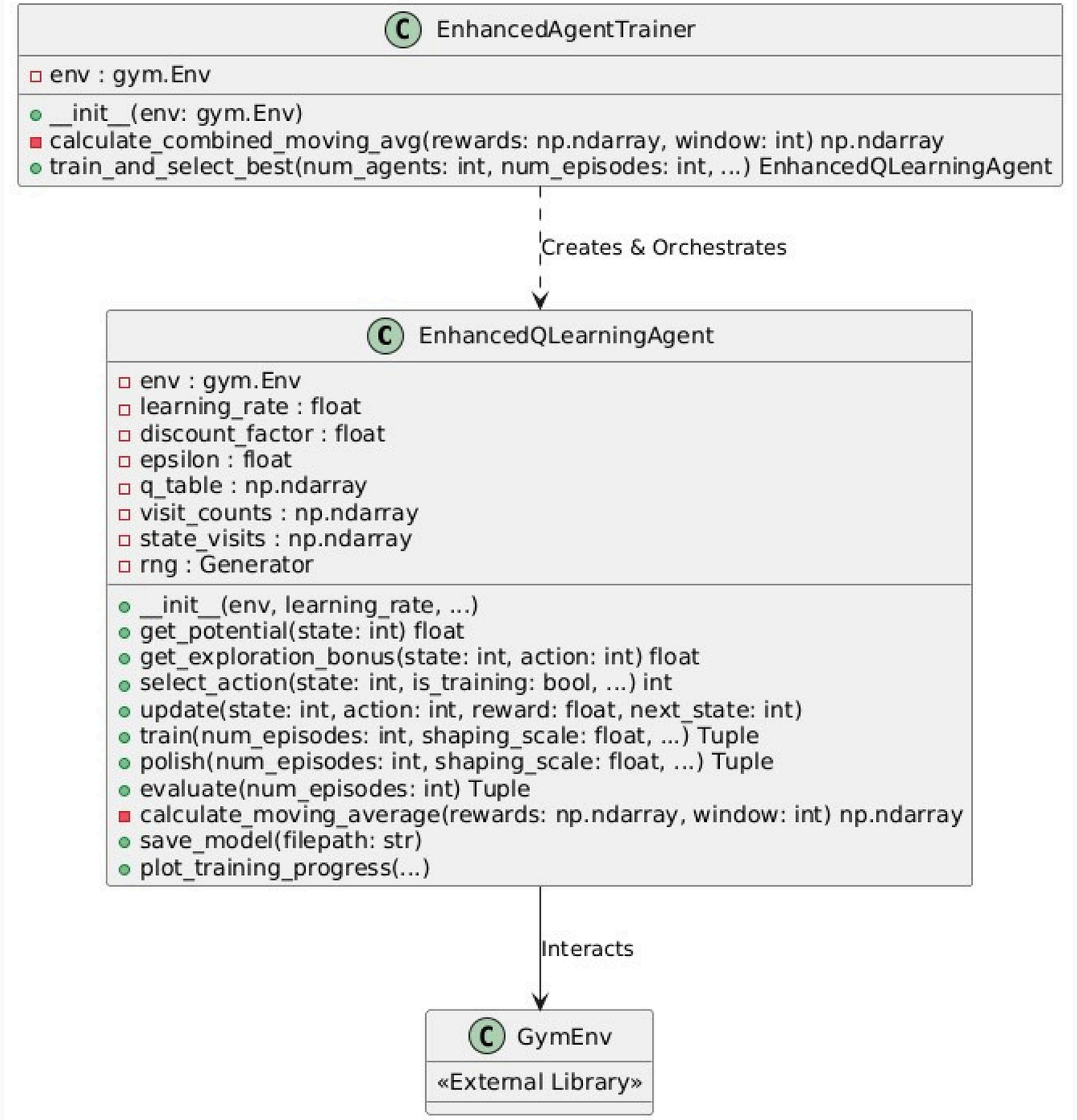
Composition:

- EnhancedAgentTrainer has-a list of EnhancedQLearningAgent instances.
- The Trainer acts as an orchestrator, managing independent worker agents.

Encapsulation:

- EnhancedQLearningAgent class encapsulates the q_table, visit_counts, and epsilon.

PART 2: UML DIAGRAM



PART 3: EXPLANATION

Goal:
Optimize a Robot (agent) that works in a Warehouse (map).
The Warehouse is divided into a rectangular grid.
A Target is randomly placed on the grid.
The Robot's goal is to reach the Target.

Agent Types:

- QLearning: Learns by using a Qtable and greedy policy
- Random: Does not learn, takes random actions

Trainer:

- In charge of training and evaluating both agents, and handles graph plotting

PART 3: OOP CONCEPTS

Abstraction:

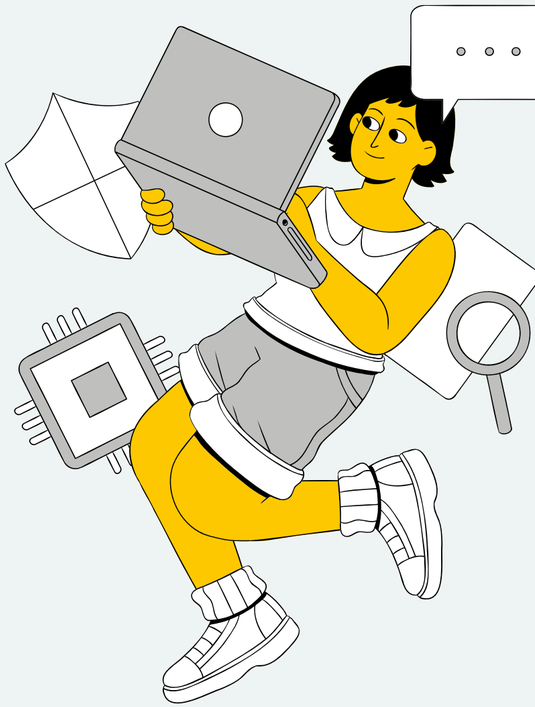
- BaseAgent defines the abstract interface: select_action(obs) and update(obs, action, reward, next_obs, done)

Composition:

- Trainer (trainer.py) has an agent and an environment
- oop_project_env has a WarehouseRobot

Polymorphism:

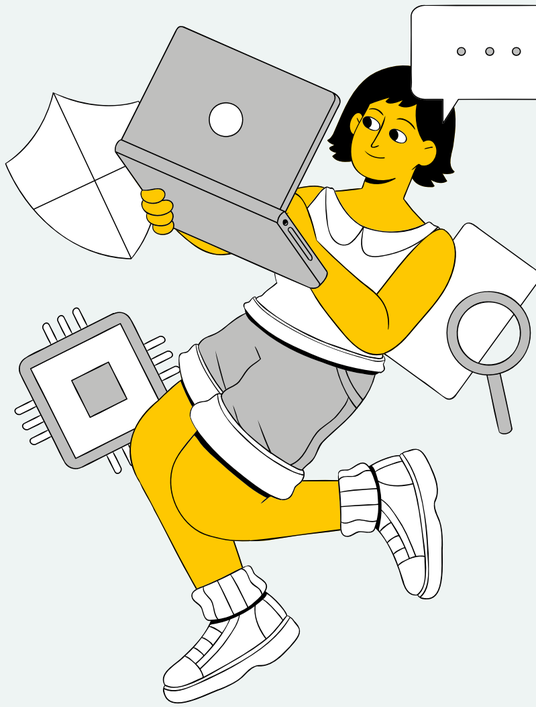
- RandomAgent and QLearningAgent inherit from the same abstract base class (BaseAgent), so they can be used wherever a BaseAgent is needed
- Trainer calls agent.select_action() and agent.update() for both RandomAgent and QLearningAgent.
- main.py chooses which agent to instantiate, but the rest of the code uses it through BaseAgent interface



PART 3: OOP CONCEPTS

Encapsulation:

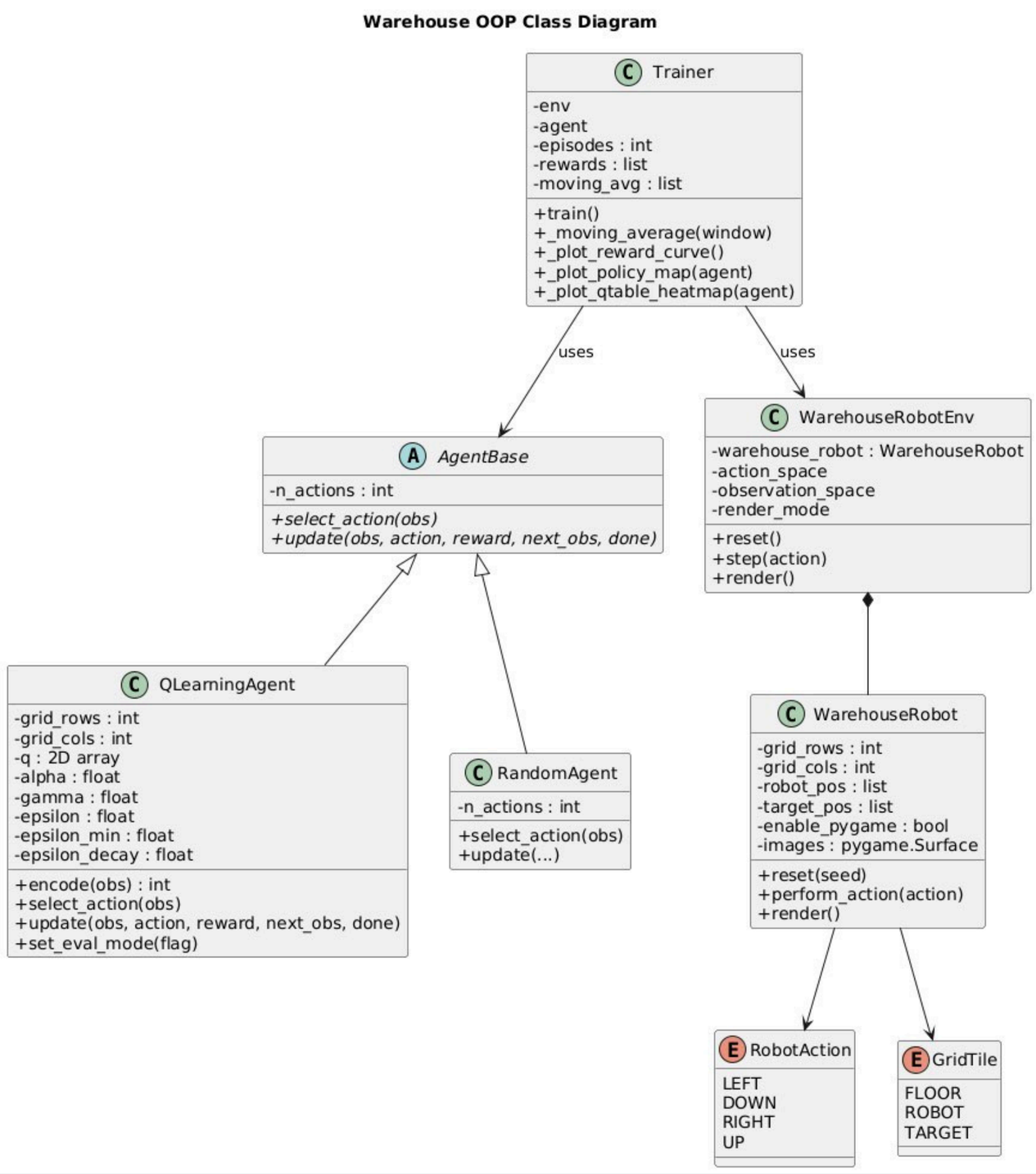
- Agents → Hide decision logic behind select_action() and update()
- QLearningAgent → Q-table, learning rate, discount factor, epsilon, decay schedule
- RandomAgent → the random action selection
- Trainer → full training loop inside trainer.train()
- WarehouseRobot → robot state (position, target) and movement/rendering logic
- oop_project_env → interaction with WarehouseRobot



Inheritance:

- RandomAgent inherits from BaseAgent, provides select_action() and update()
- QLearningAgent inherits from BaseAgent, provides Q-learning versions of select_action() and update()

PART 3: UML DIAGRAM



THANK YOU!