Lab 2

This lab has the following main goals:

1. Apply your problem-solving and software development skills using OOP in Java.
2. Boost your software design skills as you make decisions and model an agent in an environment, clearly separating the agent from the environment.
3. Practice breaking complex problems into manageable components.
4. Introduce you to implementing problems from a research paper.

Note:

- Nope, no GenAI tools allowed in this lab.
- A note on using Grammar Checker tools such as Grammarly to answer textual questions like the ones asked here: It is fine to use those as long as you DO NOT make use of generative AI features.

1. Problem Description

Modeling an agent in an environment helps solidify the OOP mindset, and that is exactly what you will do in this lab.

a) Environment (the logical entity in which agents and resources are embedded)

- You will open the paper Self-improving reactive agents based on reinforcement learning, planning and teaching.
- Observe the paper's figure 7: this is the environment you must implement.
- The environment shows enemies, food, and obstacles. However, you will treat/model enemies as obstacles. Therefore, your environment has *no* enemies.

Environment Rules.

1. Two agents cannot occupy the same cell at the same time. If an agent hits a wall or an obstacle, the agent's location remains the same.
2. Diagonal moves are forbidden.
3. If more than one agent tries to go to the same location, the environment must decide what to do. You are in charge of designing such a decision.
4. If the agent enters a location that has food, the environment rewards the agent by feeding it with points.
5. Up to you to decide and design: the number of points and whether the food resource is infinite or not.

b) Agents.

- You will create an array of n agents. Agents follow a pre-defined strategy (see below) to select actions in any of the four directions randomly.
- The environment gets an agent's current location along with its selected action and returns to the agent its next location in the environment (and food points if there are any).

Remember: you must clearly separate the concerns —those from the agent and those from the environment.

Agent - Environment Dynamics.

a. Agents start a trial in a location called "initialState". If there is more than one agent, each should have their own initial state. Design decision: should you pre-define the initial states, or pick them randomly?
b. Agents' task is to reach the environment's finalState. Multiple agents can reach the final state at the same time. You will define where the final state is.
c. Agents have four actions to choose from: {left, right, up, down}.
d. Agents select an action according to a pre-defined strategy, e.g., in an equiprobable strategy, the agent randomly picks an action and actions have the same probability of being selected.
e. At each time step, all agents select their actions. The environment processes their actions and returns a nextState in response to each agent. (Which can be the same as in a previous step.)
f. A trial ends once all agents reach the final state.
g. Trials are run in sequence. New trials are launched until a total number of trials, nTrials, is reached (feel free to use a small number, such as nTrial = 10).

5. Additional Instructions

Make sure to use OOP techniques!

Output.

You will create two tests (details below). After running the tests, your program must print, for each test:

- The environment, as shown in the paper's figure 7.
- For that environment, what is (are) the optimal number(s) of steps taken to reach the final state starting from each agents' initialState?
- How many times did each of the agents consume a food element? Overall, how many food elements were consumed?
- What is the average number of food elements taken per agent?
- What is the average number of steps taken by your agents to reach the finalState starting from the initialState?
- What is the median number of steps taken by your agents to reach the finalState starting from the initialState?

Tests in Details

You will run three agents *per* test. Agents within a test follow a strategy as follows:

1. **Test 1**, agents' strategy: {UP = 0.2, DOWN = 0.3, LEFT = 0.2, RIGHT = 0.3}.
2. **Test 2**
   a. agent 1 strategy: {UP = 0.3, DOWN = 0.2, LEFT = 0.2, RIGHT = 0.3}.
   b. Agents 2 and 3 apply an equiprobable strategy.

Hints

a) Read:
   a. Pseudorandom numbers in Java, Part 1: The background
   b. Pseudorandom numbers in Java, Part 2: Randomness with Java 17.
   c. Generating Random Numbers in Java
b) Make sure to check your random numbers.

    a. What does it mean to use the same seed?

c) Brainstorm carefully about what is part of the agent and what is part of the environment. What inputs and outputs belong to each?
d) Start with a single agent to test your approach and code.
e) Perhaps using a strategy such as: {UP = 0, DOWN = 0.5, LEFT = 0, RIGHT = 0.5} to test your approach and code could help?

6. What to hand?

You will upload a pdf file only. In the pdf, the team will answer the questions below:

1. (1.0) You will add a screenshot of your environment.
2. (1.0) Within the scope of this lab, what is an agent, and what is an environment?
3. (1.0) In your code, how did you separate the concerns of the agent from the environment's?
4. (1.5) Explain test 1 and discuss your results for that test.
5. (1.5) Explain test 2 and discuss your results for that test.
6. (1.0) Which design decisions did you make?
7. (1.0) How was the experience of making design decisions?
8. (1.0) If you were to improve the agent, what would you do?
9. (1.0) What was the most interesting thing you learned from this lab?

7. Policy.

- For team submissions, only a single submission per team. (Make sure to add all team members!!!!)

Coding Policy

1. You will read and follow our Academic Integrity Policy.
2. You will follow our course's coding policy.
3. ***Nope, you are not allowed to use AI -Powered Code Completion* Tools** in this lab.
4. Read: Can You Explain Your Code?

Grading Scheme – Each question will be graded according to the rubric below:

100%	**70%**	**60%**	**40%**	**20%**
1. All parts of the question are correctly answered. 2. Answer is comprehensive and shows great level of reflection.	1. All parts of the question are correctly answered. 2. Answer is not comprehensive.	1. Most parts of the question are correctly answered.	1. Some parts of the question are correctly answered.	1. Very few parts of the question are correctly answered.