Lab 3_A: Architecting an Adaptation to the Discrete Smart Surface benchmark

- To architect this lab, you must break down complex problems into manageable components and understand how they interact.
- Specifically, you will acquire familiarity with architecting a solution to a complex problem from a research paper.

This lab has seven goals:

1. Motivate you to grow your software design skills so that you, consequently, grow your software development skills.
2. Help enhance your skills as a **software architect**.
3. Encourage you to break down complex problems into manageable components and understand how they interact.
4. This lab requires you to convey envisioned solutions into a diagram, helping you boost your communication and spatial-thinking skills.
5. Get familiar with a research paper within the area of multi-agent systems.
6. Keep adding new entries to your Dev Journal.
7. Finally, as GenAI tools get more ubiquitous, it is key to keep up with current technologies and know *when* and *how* to use them. Therefore, part III (ONLY!) of this challenge asks you to engage with a GenAI tool of your choice.

Instructions

1) You are not allowed to split the task across team members: Everybody MUST collaboratively work on both parts.
2) Teams of max three people.
3) Remember to compile and document your thoughts into your Dev Journal.
4) You must work on part I first; move to parts II and III only after part I is concluded.
5) 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.
6) What to hand? A pdf with your responses to parts I, II, and III.

You and your peer will carefully read the Problem Description and then work on parts I and II:

1. **Part I:** Reflect on the problem description to then describe how you would architect a software solution to address that problem.
    - Your mindset must follow OOP techniques, and in Part I, we are defining *software architecture* as the blueprint of your system/solution.
    - **What to hand?** Building from your reflection, you will carefully and effectively describe how you would architect your software architecture. The description must be at least 1 page long (Tahoma, Verdana, or Palatino Linotype 12, single-spaced) and include at least one **diagram** that depicts/explains your architecture effectively.
    - To help your brainstorming, read this resource: Java Language Best Practices.

- AFTER concluding Part I, you are all set to work on Part II.

2. **Part II:** In Career Readiness 1, the resources pushed you to reflect on AI agent orchestration. Now, let's use *parallel* and *systems thinking skills* to answer:
    1. In what ways can you connect the concept of AI orchestration with this lab?
    2. How can this lab help you think about designing processes for multiple agents and orchestrating them?
    3. Continuing from that reflection, consider edge cases and what can go wrong. Going even further, in what ways can the *Ariane 5 launch accident* inform your reflection on what can go wrong when orchestrating agents?
    - Answers must be comprehensive (Tahoma, Verdana, or Palatino Linotype 12)
3. **Part III:** You will write a prompt to brainstorm with two GenAI tools of choice.
   IMPORTANT: brainstorm with a GenAI tool, which is different from feeding it with a prompt that guides it to a final answer/solution to the problem.
    a) Your prompt must: summarize the problem description and ask the tool to help you brainstorm software architecture to solve that problem. That means you will ask it a few questions as you brainstorm.
        - In part II, you can either interpret *software architecture* as a blueprint or as roadmap. **Make sure to state**, in your reflection, what definition you are using for *software architecture*.

    b) Which GenAI tools should I use? You are free to choose. To help you select, take a look at: Software Architect and this thread on reddit.
        - Important: you are NOT allowed to feed this deliverable into the GenAI tool, nor your own response to Part I.

    c) The part III reflection must revisit your response for part I and describe how your interaction with the GenAI tools helped you rethink your software architecture.
        - It should also compare your experience/impression from interacting with two GenAI tools to brainstorm on the same problem: which one did you prefer and why?
        - Finally, it must elaborate on: what did you add from this lab into your Dev Journal and why?
        - Part III must be at least 1-page long (Tahoma, Verdana, or Palatino Linotype 12)

    d) **What to hand?** Your prompt followed by a second reflection, a revisited diagram, and details of what GenAI tools you interacted with.
        - Remember to provide the definition you used for *software architecture*.

Problem Description.

- You will work with a simplification of the Discrete Smart Surface benchmark (reference: Independent reinforcement learners in cooperative Markov games: a survey regarding coordination)
- We described the benchmark here: Situated Learners in a Sequential Decision-Making Setting. This Programming Challenge's problem description is actually a simplification of this work.

Problem in details:

1. There is a 2D grid of N x M situated agents (see figure 1 below). At each time step, all agents individually pick actions to move an object.
    a. Agents individually pick their actions and feed them to the environment.
    b. *A* is the set of actions available to each agent, and A = {left, right, up, down, still}.
    c. For the purposes of this assignment, agents pick their actions randomly.
2. To accomplish the task, agents must successfully coordinate actions so that the object moves from the initial position, depicted by i in the image below, to the goal or terminal location g.
    a. A trial starts from placing the object at the initial position i and finishes once the object's top left reaches the goal state g. The object does not rotate and cannot leave the surface.
    b. A motion of the 2D object that is placed on the 2D grid's surface is determined by the Environment using a weighted sum of the agents' actions.
    c. Agents are dynamically categorized into three groups: agents by the object, agents under the object, and remaining agents. Each group has a weight that impacts the motion dynamics: wb, wu, ws, respectively, and {wb = 1, wu = 5, ws = 0}. If there is more than one action with the highest number, the environment chooses randomly among those. That process results in the *applied_action*.
3. The environment applies the *applied_action* so that the object moves from the *current_state* to the *next_state.* (If the *applied_action* = "stay still", even though the object does not move, there is a state transition.)
4. *E* is the width of the grid's right and left borders, which are penalty areas. Borders have equal width and height, such as 1 × N each.
    a. If the object reaches a penalty area, all agents continuously get a punishment of = -10 until the object moves to a non-penalty area. Otherwise, agents continuously receive a score of -1, independently of the object's motion. Finally, once the object reaches the goal/terminal location, all agents receive a score of 100, and the trial ends.
5. *S* is the set of states. Each possible object's position defines a unique state; hence, there are (M −m+1)(N −n+1) states, see figure 2.

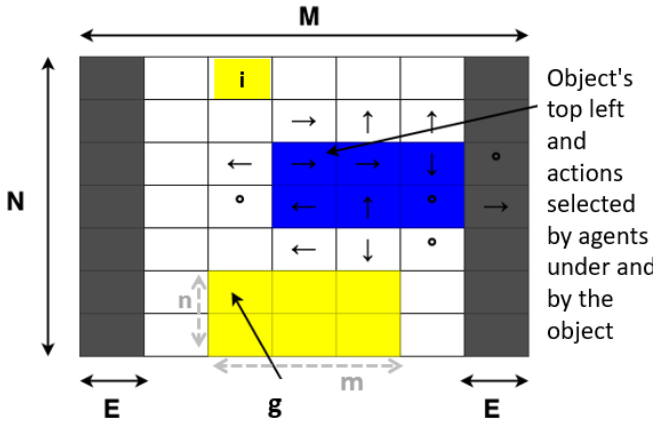

Figure 1. Each cell corresponds to a situated agent. The agents' task: to move the object from an initial position i into the goal state g (which defines the terminal state). The 3 × 2 object is color-coded in blue. Arrows inside the grid correspond to actions selected by agents under or by the object (circles for 'stay still'). The leftmost and rightmost columns are penalty areas, which agents also occupy.

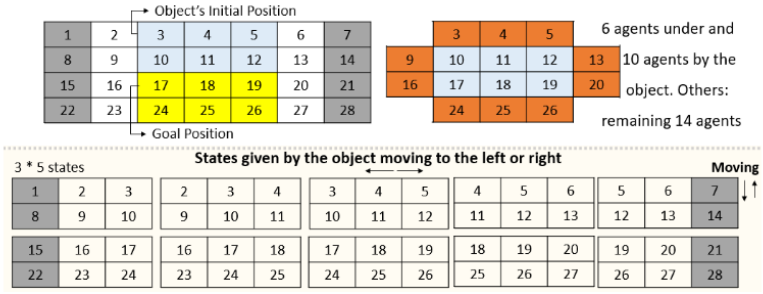

Figure 2. Upper left: a toy example; numbers within cells represent a situated agent. Upper right: a state in which 16 agents' actions play a role in the transition function F; remaining 14 agents are ignored ($w_s = 0$). Lower level: 15 possible states given by the grid and the object's dimensions (intermediary row omitted).

Figure 2, the lower part shows the states that follow as the object moves: five columns sideways by three rows in the up/down directions. Within the states, we show the corresponding agents that would be under the object. At each time step, the object's top-left provides s^t, the state s at the time step t.

Figure 2, upper right, depicts a state with the following agents under the object: 10-12 and 17-19. Now, see the interplay between state and agent role/importance: in this state, only 16 out of 28 agents play a role (agents by and under the object only). Hence, Figure 2 helps notice that the number of agents by the object varies as the object moves: for this example, 5 agents if at the corners, 7 if at the top or bottom states, 8 sideways, and 10 otherwise – whereas the number of agents under the object remains constant.

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

- For each part, it will be considered if you:
  - Followed the instructions.
  - Provided a comprehensive and meaningful explanation.
  - Provided a diagram along with a comprehensive caption.