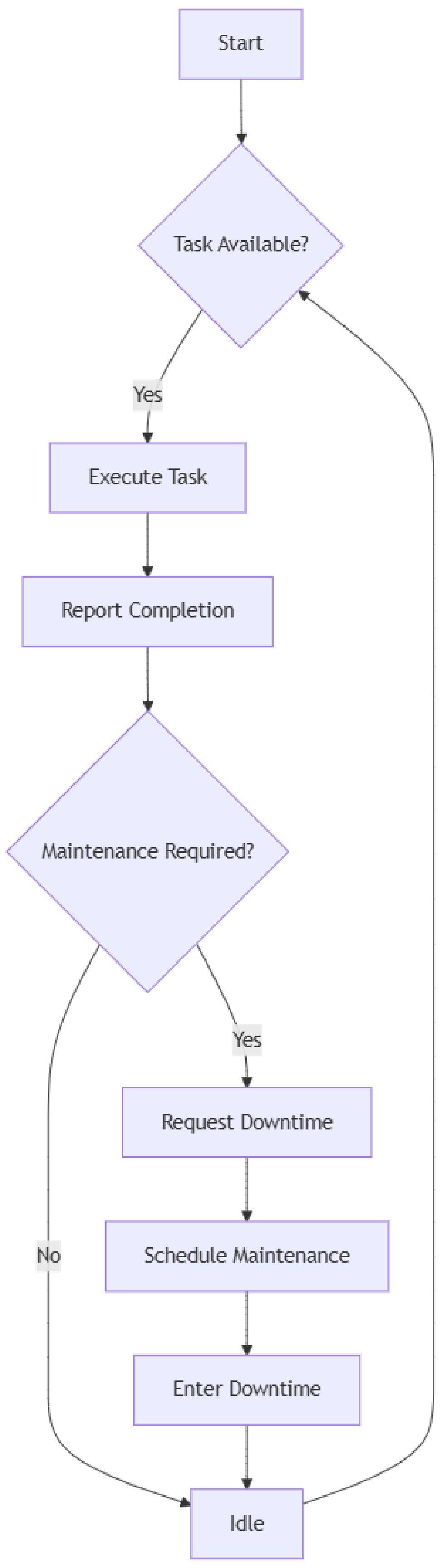
UML Design

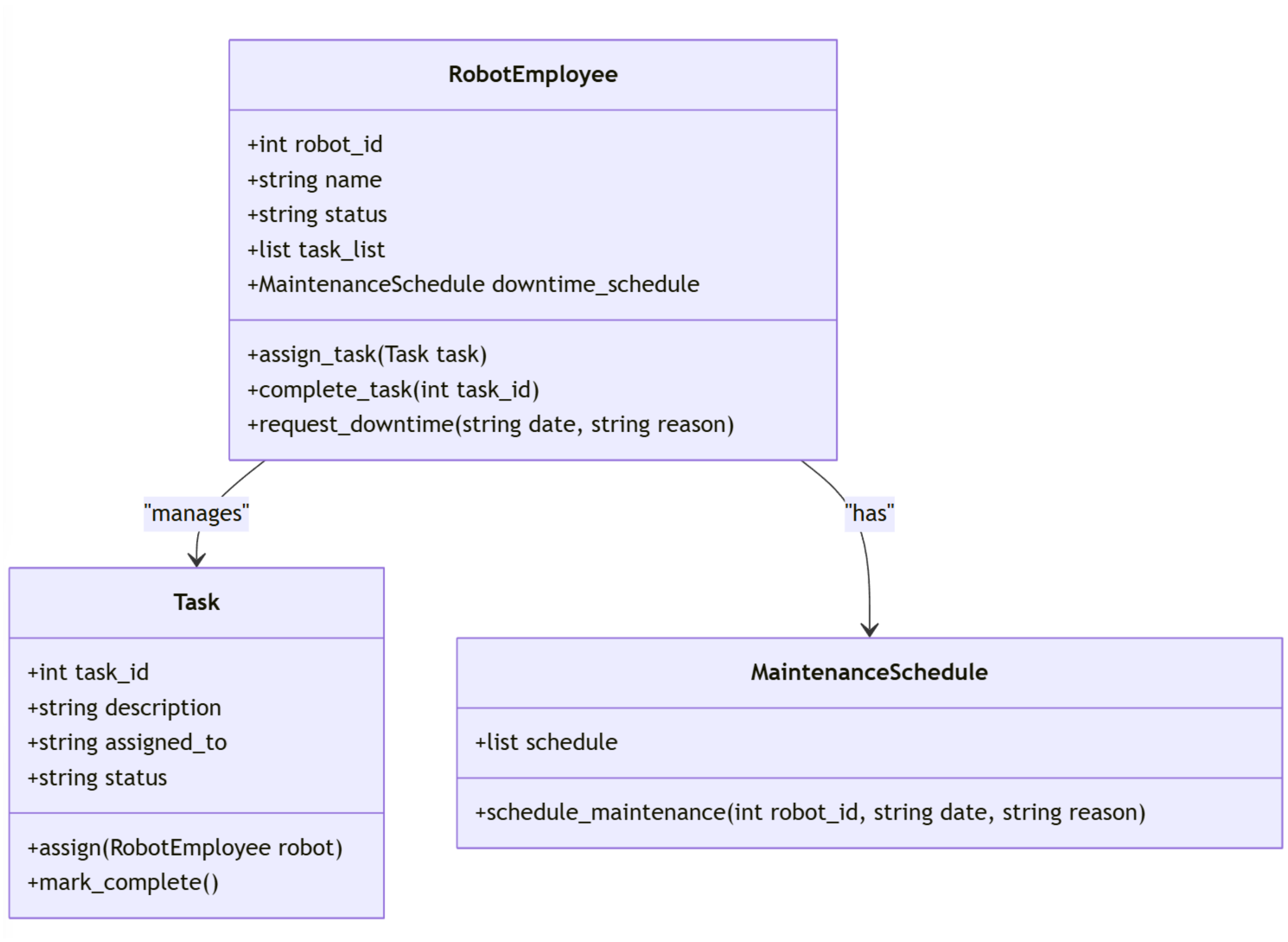
Task One

Design an activity diagram which shows the relationships and interactivity between the tasks executed by a humanoid robot performing a task of your choice.



Task Two

Expand the activity diagram with the development of a class diagram using UML to support a system with basic employee-related functionality. This should include the retention of employee details and allow an employee to book a day of annual leave.



Task Three

Develop the Python program to implement the class model.

```python
class Task:
    def __init__(self, task_id, description):
        self.task_id = task_id
        self.description = description
        self.assigned_to = None
        self.status = "Pending"

    def assign(self, robot):
        self.assigned_to = robot.robot_id
        self.status = "In Progress"

    def mark_complete(self):
        self.status = "Completed"


class MaintenanceSchedule:
    def __init__(self):
        self.schedule = []

    def schedule_maintenance(self, robot_id, date, reason):
        self.schedule.append({
            "robot_id": robot_id,
            "date": date,
            "reason": reason
        })


class RobotEmployee:
    def __init__(self, robot_id, name):
        self.robot_id = robot_id
        self.name = name
        self.status = "Idle"
        self.task_list = []
        self.downtime_schedule = MaintenanceSchedule()

    def assign_task(self, task):
        task.assign(self)
        self.task_list.append(task)
        self.status = "Busy"

    def complete_task(self, task_id):
        for task in self.task_list:
            if task.task_id == task_id:
                task.mark_complete()
                break
        if all(task.status == "Completed" for task in self.task_list):
            self.status = "Idle"

    def request_downtime(self, date, reason):
        self.downtime_schedule.schedule_maintenance(self.robot_id, date, reason)
        self.status = "In Maintenance"


# Example Usage
if __name__ == "__main__":
    # Create robots
    robot1 = RobotEmployee(1, "RoboHelper")

    # Create tasks
    task1 = Task(101, "Pick up package")
    task2 = Task(102, "Deliver package")

    # Assign tasks
    robot1.assign_task(task1)
```

```python
robot1.assign_task(task2)

# Complete a task
robot1.complete_task(101)

# Request downtime for maintenance
robot1.request_downtime("2025-01-15", "Battery recharge")

# Output robot and task statuses
print(f"Robot {robot1.name} Status: {robot1.status}")
for task in robot1.task_list:
    print(f"Task {task.task_id}: {task.status}")

print("Maintenance Schedule:", robot1.downtime_schedule.schedule)
```