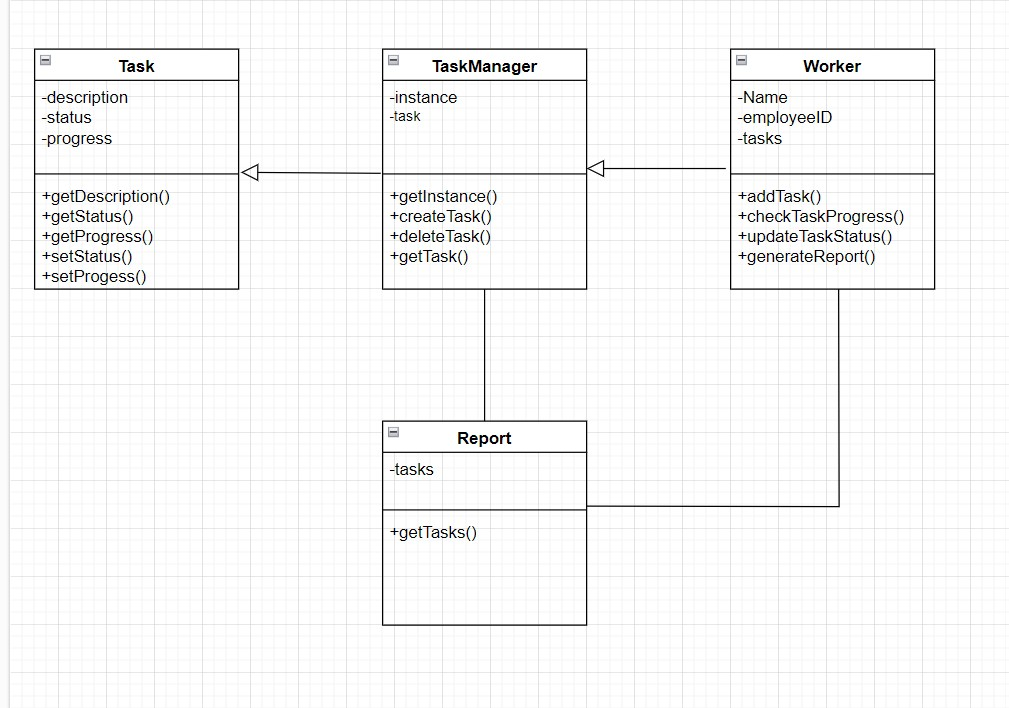
1. Lập trình hướng đối tượng (OOP):
   * Sử dụng lớp "Worker" để đại diện cho mỗi công nhân. Lớp này có thể chứa các thuộc tính như tên, mã nhân viên và danh sách công việc đang thực hiện.
   * Sử dụng lớp "Task" để đại diện cho mỗi công việc. Lớp này có thể chứa các thuộc tính như mô tả công việc, trạng thái và tiến độ.
   * Sử dụng lớp "TaskManager" để quản lý các công việc. Lớp này có thể chứa các phương thức để tạo, xóa và cập nhật công việc.
2. Mẫu thiết kế Singleton:
   * Áp dụng mẫu thiết kế Singleton cho lớp "TaskManager" để đảm bảo rằng chỉ có một đối tượng "TaskManager" duy nhất trong hệ thống. Điều này giúp đảm bảo rằng quản lý công việc được truy cập và sử dụng một cách nhất quán trong toàn bộ ứng dụng.
3. Mẫu thiết kế Observer:
   * Sử dụng mẫu thiết kế Observer để thông báo cho công nhân về tiến độ của các công việc. Khi trạng thái hoặc tiến độ của một công việc thay đổi, hệ thống thông báo cho công nhân tương ứng thông qua cơ chế Observer. Điều này giúp công nhân cập nhật và theo dõi công việc một cách tự động.
4. Mẫu thiết kế Strategy:
   * Sử dụng mẫu thiết kế Strategy để cho phép công nhân lựa chọn phương pháp xử lý công việc theo nhu cầu cụ thể. Ví dụ, công nhân có thể chọn một chiến lược (strategy) để ưu tiên các công việc theo tiến độ hoặc ưu tiên các công việc theo độ ưu tiên. Điều này giúp linh hoạt trong quá trình quản lý công việc.

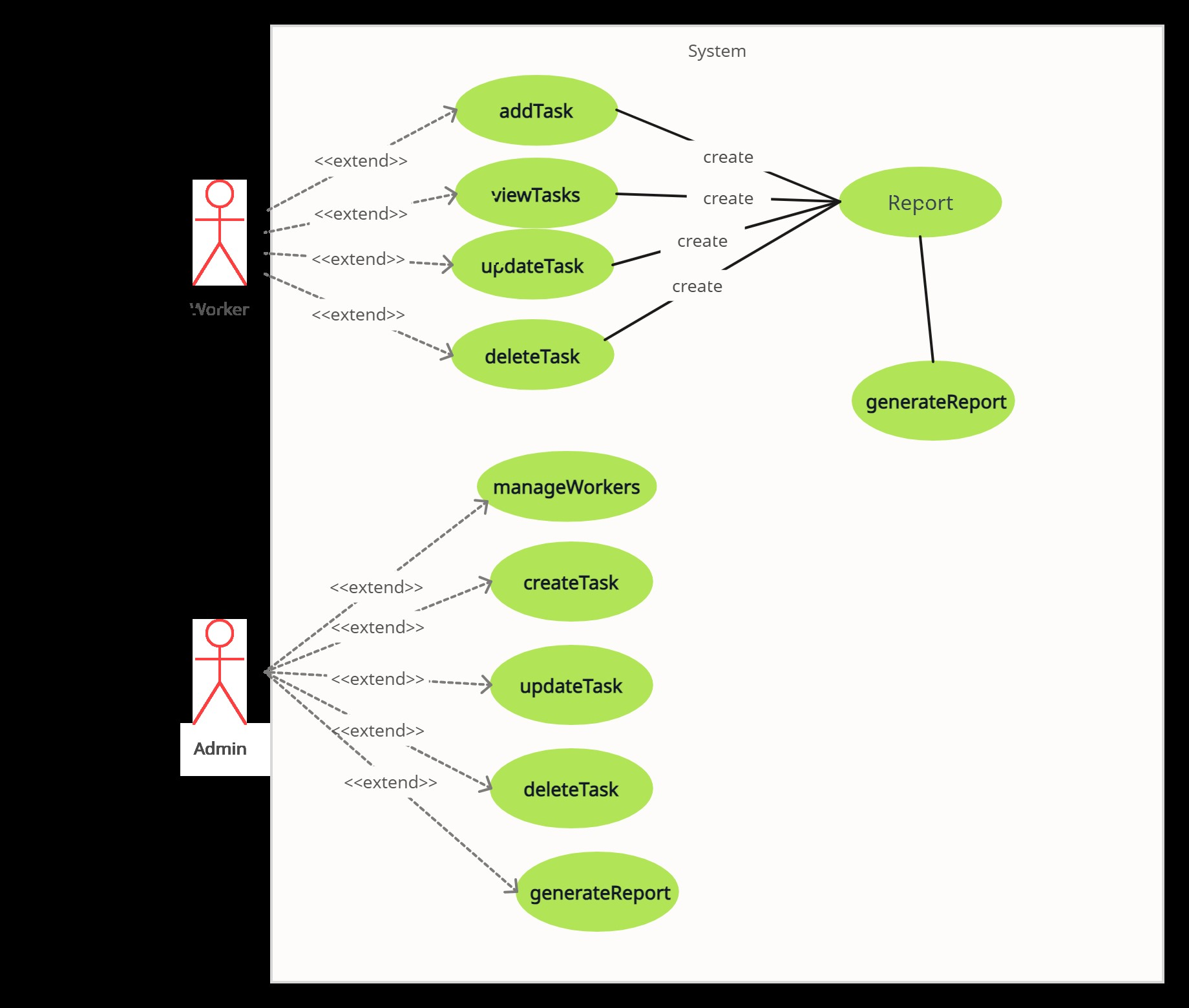
Tóm lại, OOP và một số mẫu thiết kế như Singleton, Observer và Strategy có thể được áp dụng trong quản lý công nhân để tăng tính linh hoạt, tái sử dụng và hiệu suất của hệ thống. Việc sử dụng các khái niệm và mẫu thiết kế này giúp xây dựng một hệ thống quản lý công việc hiệu quả và dễ dàng mở rộng trong tương lai.

Chúng ta sẽ tạo một hệ thống quản lý công việc cho công nhân trong một nhà máy sản xuất. Hệ thống này sẽ cho phép công nhân quản lý các công việc, kiểm tra tiến độ, và tạo báo cáo.

1. Diagram lớp (Class diagram):
   * Lớp "Worker" (Công nhân): Đại diện cho công nhân, có các thuộc tính như tên, mã nhân viên và danh sách các công việc đang thực hiện.
   * Lớp "Task" (Công việc): Đại diện cho một công việc cần thực hiện, có các thuộc tính như mô tả công việc, trạng thái (hoàn thành/chưa hoàn thành) và tiến độ.
   * Lớp "TaskManager" (Quản lý công việc): Đại diện cho trung tâm quản lý công việc, có chức năng tạo, xóa và cập nhật các công việc. Nó cũng giữ danh sách các công việc hiện có và cung cấp phương thức để công nhân thực hiện các thao tác như kiểm tra tiến độ và tạo báo cáo.
2. Use case (Mô tả các chức năng hệ thống):
   * Use case 1: Tạo công việc
     + Công nhân gửi yêu cầu tạo công việc mới cho Quản lý công việc.
     + Quản lý công việc tạo một đối tượng "Task" mới và thêm vào danh sách công việc hiện có.
   * Use case 2: Kiểm tra tiến độ công việc
     + Công nhân chọn một công việc từ danh sách công việc hiện có.
     + Hệ thống hiển thị tiến độ của công việc đó.
   * Use case 3: Cập nhật trạng thái công việc
     + Công nhân chọn một công việc từ danh sách công việc hiện có.
     + Công nhân cập nhật trạng thái của công việc (hoàn thành/chưa hoàn thành).
     + Hệ thống cập nhật trạng thái của công việc trong danh sách.

* Use case 4: Tạo báo cáo công việc
* Công nhân yêu cầu tạo báo cáo công việc.
* Hệ thống truy cập danh sách công việc và thu thập thông tin về các công việc đã hoàn thành.
* Hệ thống tạo một báo cáo chứa thông tin về tiến độ và trạng thái của các công việc.
* Hệ thống hiển thị báo cáo cho công nhân.





**Giải thích :**

Trong biểu đồ use case của hệ thống quản lý công việc, chúng ta có các thành phần sau:

1. Actor (Tác nhân):
   * Worker (Công nhân): Đại diện cho nhân viên thực hiện các công việc trong hệ thống.
   * Admin (Quản trị viên): Đại diện cho người quản lý và điều hành hệ thống.
2. Use Case (Use Case):
   * Add Task (Thêm công việc): Use case mô tả quá trình thêm công việc mới vào hệ thống.
   * View Tasks (Xem danh sách công việc): Use case mô tả quá trình xem danh sách công việc hiện có trong hệ thống.
   * Update Task (Cập nhật công việc): Use case mô tả quá trình cập nhật thông tin chi tiết của một công việc.
   * Delete Task (Xóa công việc): Use case mô tả quá trình xóa một công việc khỏi hệ thống.
   * Manage Workers (Quản lý công nhân): Use case mô tả quá trình quản lý danh sách công nhân trong hệ thống.
   * Create Task (Tạo công việc): Use case mô tả quá trình tạo công việc mới cho các công nhân trong hệ thống.
   * Generate Report (Tạo báo cáo): Use case mô tả quá trình tạo báo cáo về công việc đã hoàn thành trong hệ thống.
3. Mối quan hệ:
   * Extend (Kế thừa): Mối quan hệ extend được sử dụng để chỉ ra rằng một use case có thể mở rộng hoặc mở rộng các chức năng của một use case khác. Ví dụ: **addTask()** có thể mở rộng bởi **viewTasks()**, cho phép công nhân xem danh sách công việc sau khi thêm công việc mới.

**Vai trò của OOP và DP**

1. Vai trò của OOP trong sơ đồ:
   * OOP giúp xác định cách tổ chức và sắp xếp các đối tượng trong hệ thống. Nó tập trung vào việc xây dựng các lớp, đối tượng và mối quan hệ giữa chúng.
   * OOP cho phép đóng gói dữ liệu và phương thức vào trong các đối tượng, giúp tăng tính bảo mật và ẩn thông tin.
   * OOP cung cấp tính kế thừa, cho phép tái sử dụng mã nguồn và xây dựng các mối quan hệ phụ thuộc giữa các lớp.
   * OOP tạo điều kiện cho việc triển khai các khái niệm như đa hình, trừu tượng hóa và đóng gói, giúp mô hình hóa thế giới thực vào trong hệ thống.
2. Vai trò của DP trong sơ đồ:
   * DP cung cấp các mô hình, tiêu chuẩn và hướng dẫn để giải quyết các vấn đề phổ biến trong thiết kế hệ thống.
   * DP giúp giảm thiểu sự phụ thuộc giữa các thành phần của hệ thống, tạo ra sự linh hoạt và dễ dàng mở rộng.
   * DP giúp tăng tính module hóa của hệ thống, cho phép thay đổi và nâng cấp các thành phần một cách dễ dàng mà không ảnh hưởng đến các thành phần khác.
   * DP cung cấp các giải pháp kiến trúc, cấu trúc và thiết kế thông qua các mẫu đã được chứng minh và kiểm chứng để giải quyết các vấn đề phức tạp trong thiết kế hệ thống.