Here's a structured approach for implementing the Pager class in TypeScript using queues to manage engineers and managers:

Approach:

Use Two Queues: One for engineers and another for managers.

Incident Assignment:

Assign incidents to engineers first.

If all engineers are busy, assign to managers.

If no one is available, keep the incident in a pending queue.

Freeing Up:

When an engineer or manager becomes available, they pick from the pending incidents.

Implementation:

Pager.ts

typescript

Copy

Edit

class Pager {

private engineers: string[]; // Queue of engineers

private managers: string[]; // Queue of managers

private busy: Set<string>; // Set to track busy engineers/managers

private pendingIncidents: string[]; // Queue of pending incidents

constructor(engineers: string[], managers: string[]) {

this.engineers = [...engineers];

this.managers = [...managers];

this.busy = new Set();

this.pendingIncidents = [];

}

// Assign incident to the first available person

assignIncident(incident: string): string {

for (const list of [this.engineers, this.managers]) {

for (const person of list) {

if (!this.busy.has(person)) {

this.busy.add(person);

return ${incident} assigned to ${person};

}

}

}

// If no one is available, add to pending queue

this.pendingIncidents.push(incident);

return ${incident} is pending, no available personnel;

}

// Mark a person as available and assign any pending incident

resolveIncident(person: string): string {

if (!this.busy.has(person)) {

return ${person} is not currently handling an incident;

}

this.busy.delete(person);

// Assign a pending incident if available

if (this.pendingIncidents.length > 0) {

const incident = this.pendingIncidents.shift()!;

this.busy.add(person);

return ${incident} assigned to ${person};

}

return ${person} is now available;

}

}

// Example Usage

const pager = new Pager(["Alice", "Bob"], ["Charlie"]);

console.log(pager.assignIncident("Incident1")); // Assigns to Alice

console.log(pager.assignIncident("Incident2")); // Assigns to Bob

console.log(pager.assignIncident("Incident3")); // Assigns to Charlie

console.log(pager.assignIncident("Incident4")); // Pending

console.log(pager.resolveIncident("Alice")); // Assigns pending incident to Alice

Test Cases:

Pager.test.ts

typescript

Copy

Edit

import { Pager } from "./Pager";

describe("Pager Duty System", () => {

let pager: Pager;

beforeEach(() => {

pager = new Pager(["Alice", "Bob"], ["Charlie"]);

});

test("assign incidents to engineers first", () => {

expect(pager.assignIncident("Incident1")).toBe("Incident1 assigned to Alice");

expect(pager.assignIncident("Incident2")).toBe("Incident2 assigned to Bob");

});

test("assign incidents to managers if engineers are busy", () => {

pager.assignIncident("Incident1");

pager.assignIncident("Incident2");

expect(pager.assignIncident("Incident3")).toBe("Incident3 assigned to Charlie");

});

test("queue incidents when all personnel are busy", () => {

pager.assignIncident("Incident1");

pager.assignIncident("Incident2");

pager.assignIncident("Incident3");

expect(pager.assignIncident("Incident4")).toBe("Incident4 is pending, no available personnel");

});

test("resolve incidents and assign pending ones", () => {

pager.assignIncident("Incident1");

pager.assignIncident("Incident2");

pager.assignIncident("Incident3");

pager.assignIncident("Incident4"); // This one is pending

expect(pager.resolveIncident("Alice")).toBe("Incident4 assigned to Alice");

});

});

Key Features:

✅ Uses queues for engineers, managers, and pending incidents

✅ Assigns incidents to engineers first, then managers

✅ Handles pending incidents when someone becomes available

✅ Unit tests included for validation

Would you like any modifications or additional functionality? 🚀