Analisi e Progettazione del Software

Prova scritta del 7 febbraio 2019

Si vuole progettare una classe C++ per la gestione giornaliera di un reparto chirurgico di un ospedale. Per semplicità si assuma che le operazioni durino tutte un'ora e che vengano sempre schedulate alle ore esatte (quindi l'orario è rappresentabile tramite un intero).

Le operazioni principali per il reparto, di cui si memorizza il nome e l'orario di apertura (anch'esso valore intero), sono le seguenti:

InserisciSala($s : Sala$)
La sala operatoria s viene inserita tra quelle disponibili per il reparto. La sala è registrata come *generica*, cioè non è dedicata ad alcuna specialità.
Precondizioni: La sala s non è già stata inserita.

RegistraPaziente($p : Paziente$)
Il paziente p viene registrato tra quelli da operare per la giornata, ma la sua operazione non viene assegnata ad alcuna sala oparatoria.
Precondizioni: Il paziente p non è già stato registrato.

DedicaSala($s : Sala, sp : stringa$)
La sala s viene dedicata ad operazioni della specialità di nome sp.
Precondizioni: La sala s è già stata inserita nel reparto.

AssegnaPaziente($p : Paziente, s : Sala, o : intero, u : booleano$)
Il paziente p viene assegnato alla sala s. Se l'assegnazione del paziente è urgente (u vero) allora viene assegnato nell'orario o e l'eventuale paziente in orario o viene posticipato di un'ora (eventualmente posticipando altri pazienti in cascata). Se invece l'assegnazione non è urgente (u falso), allora il paziente viene assegnato alla prima ora libera partendo da o.
Precondizioni: Il paziente è già registrato nel reparto. L'orario o è maggiore o uguale all'apertura del reparto. Se l'assegnazione del paziente non è urgente, la sala deve essere dedicata alla specialità dell'operazione del paziente p, oppure non dedicata ad alcuna specialità (se il paziente è urgente può essere assegnato a qualsiasi sala).

RimuoviPaziente($p : Paziente$)
Il paziente p viene rimosso dal reparto. Se il paziente è assegnato ad una sala, anche la sua assegnazione viene eliminata.
Precondizioni: Il paziente p è registrato.

Esercizio 1 (punti 4) Si disegni il diagramma UML delle classi per l'applicazione.

Esercizio 2 (punti 5) Si scriva la definizione della classe `Reparto` e delle eventuali altre classi che compongono il diagramma UML, assumendo però già disponibili e *immodificabili* le classi `Paziente` e `Sala` per la gestione di pazienti e sale operatorie, di cui non si conosce la rappresentazione. Si assuma soltanto che il paziente abbia il metodo `Spec()` che restituisce la stringa che rappresenta la specialità della sua operazione.

Esercizio 3 (punti 11) Si scrivano le definizioni dei metodi della classe `Reparto` che corrispondono alle operazioni sopra elencate e i selettori che si ritengono opportuni. Si gestiscano le precondizioni tramite il lancio dell'eccezione `invalid_argument`. Si definiscano i metodi (modificatori e selettori) delle eventuali altre classi del diagramma UML (escluse ovviamente `Paziente` e `Sala`).

Esercizio 4 (punti 4) Si scriva l'operatore di output della classe `Reparto`, in modo che stampi anche tutti i dati ad essa collegati. Si assumano disponibili gli operatori di output delle classi `Paziente` e `Sala`.

Esercizio 5 (punti 6) Si scriva una funzione esterna (non *friend*) alla classe `Reparto` che riceva come parametro un oggetto della classe `Reparto` ed una stringa che rappresenta una specialità e restituisca il primo orario libero per un'operazione per quella specialità (quindi in una sala dedicata a quella specialità oppure generica).