Analisi e Progettazione del Software
Prova scritta del 21 gennaio 2020

Si intende progettare un'applicazione per la gestione dei ricoveri di un reparto ospedaliero. L'applicazione riguarda i pazienti e le stanze del reparto. Di ciascuna stanza interessa il numero di letti e il tipo. Esistono quattro tipi di stanza: solo_donne, solo_uomini, singolo_genere e misto. Nelle stanze di tipo solo_donne (risp. solo_uomini) sono ammesse solo donne (risp. solo uomini). Nelle stanze di tipo singolo_genere possono esserci solo pazienti uomini *oppure* solo pazienti donne, mentre nelle stanze di tipo misto ci possono anche essere uomini e donne contemporaneamente. Per ciascun ricovero nel reparto interessano il paziente, la stanza in cui avviene e la data presunta di dimissione del paziente. Un paziente si dice *compatibile* con una stanza se può essere inserito nella stanza senza violare le regole di genere della stanza, rispetto ai pazienti già presenti.

Si considerino le seguenti operazioni per la classe Reparto:

AggiungiStanza($c : intero, ts : tipo\_stanza$)
Viene aggiunta al reparto una stanza di capacità c e di tipo ts.

AmmettiPaziente($p : Paziente, d : Data$) : *booleano*
Il paziente p viene ricoverato con data di dimissione presunta d. In particolare, il paziente p viene assegnato alla prima stanza compatibile che abbia un posto libero. Nel caso non vi sia alcuna stanza in tali condizioni, la funzione non esegue alcuna assegnazione e restituisce $false$ (restituisce invece $true$ in caso positivo).
Precondizione: il paziente p non è già ricoverato nel reparto.

DimettiPaziente($p : Paziente$)
Il paziente p viene dimesso e quindi eliminato dal reparto.
Precondizione: il paziente p è ricoverato in una delle stanze del reparto.

IsolaPaziente($p : Paziente, k : intero$) : *booleano*
Il paziente p viene spostato nella prima stanza compatibile vuota. Se non c'è una stanza compatibile vuota oppure se il paziente è già in stanza da solo, allora il paziente non viene spostato e viene restituito $false$ (viene restituito $true$ se avviene lo spostamento). In ogni caso, la data di dimissione presunta viene incrementata di k giorni.
Precondizione: il paziente p è ricoverato in una delle stanze del reparto.

Si assuma già disponibile la classe `Paziente` con un selettore per il genere, che restituisce un valore del tipo `enum class Genere {FEMMINA, MASCHIO}`, e con l'operatore di output. Si assuma già disponibile anche la classe `Data` con tutti i metodi e gli operatori che si ritengono opportuni.

Esercizio 1 (punti 4) Si disegni il diagramma UML delle classi per l'applicazione, che comprenda anche le responsabilità delle associazioni.

Esercizio 2 (punti 5) Si scrivano le definizioni delle classi C++ per l'applicazione (escluse quelle già disponibili).

Esercizio 3 (punti 12) Si scrivano le definizioni dei metodi che corrispondono alle operazioni sopra elencate, gestendo le precondizioni tramite lancio dell'eccezione `invalid_argument`.

Esercizio 4 (punti 4) Si scriva l'operatore di output della classe che rappresenta il reparto, in modo che stampi anche tutti i dati ad essa collegati.

Esercizio 5 (punti 5) Si scriva una funzione esterna (non *friend*) che riceva come parametri un oggetto `r` di classe `Reparto` e una data `d`. La funzione deve restituire il numero di stanze di `r` che saranno completamente libere alla data `d` (cioè tali che non siano occupate da alcun paziente con una data di dimissione presunta maggiore di `d`).