Analisi e Progettazione del Software
Prova scritta del 29 giugno 2018

Si vuole progettare una classe C++ per la gestione di un'agenzia turistica che organizza visite a monumenti.

Le operazioni principali per l'agenzia, di cui si memorizza il nome, sono le seguenti:

- Inserimento di un momumento nell'offerta dell'agenzia (si ignori invece la cancellazione di monumenti dall'offerta).
- Inserimento di un nuovo cliente (si ignori anche qui la cancellazione).
- Registrazione dell'associazione tra clienti e monumenti, che rappresenta l'interesse di un cliente per la visita di un momumento, e di cui bisogna memorizzare anche l'orario preferito. Ovviamente un cliente può avere interesse per più monumenti.
- Creazione di una visita: dati come parametri in ingresso un momumento ed un orario, viene creata una nuova visita, per la quale viene anche inserito un codice alfabetico (passato anch'esso come parametro). Vengono automaticamente inserite nella visita tutte le persone che avevano espresso l'interesse per quel momumento per quell'ora. Il corrispondente interesse viene cancellato (in quanto soddisfatto dalla visita).
- Inserimento di una persona in una visita. Dato il codice di una visita ed una persona, la persona viene inserita nella visita indipendentemente dal fatto che abbia espresso o meno interesse per il monumento della visita. Nel caso in cui tale interesse fosse stato espresso, questo viene calcellato (qualunque sia l'orario).

Esercizio 1 (punti 7) Si disegni il diagramma UML delle classi per l'applicazione.

Esercizio 2 (punti 6) Si scrivano le definizioni della classe `Agenzia` e delle altre classi che compongono il diagramma UML, assumendo però già disponibili e immodificabili le classi `Persona` e `Monumento` per la gestione di persone e monumenti, di cui però non si conosce la rappresentazione.

Si consideri altresì disponibile la classe `Orario` che memorizza un orario della giornata. Si assuma che la classe `Orario` metta a disposizione tutti i metodi e gli operatori che si ritengono opportuni.

Esercizio 3 (punti 13) Si scrivano le definizioni dei metodi della classe `Agenzia` che corrispondono alle operazioni sopra elencate e i selettori che si ritengono opportuni. Si definiscano delle opportune precondizioni, che andranno gestite tramite il lancio dell'eccezione `invalid_argument`. Si definiscano i metodi (modificatori e selettori) delle altre classi del diagramma UML (escluse ovviamente `Persona` e `Monumento`).

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