Progetti II Semestre
Parte 1: Strutture Dati

Programmazione e Analisi di Dati
Mod. A – Programmazione Java

Ultimo aggiornamento: 10 Marzo 2015

La parte del corso che si tiene nel secondo semestre prevede lo svolgimento di due progetti. In questo documento è descritta una lista di possibili progetti tra i quali scegliere il **PRIMO** dei due da svolgere.

1. Progetti di applicazione di quanto visto a lezione (per max 2 persone)

Progetto 1.1.1 Realizzare una classe `AsiloNido` che consente di gestire un insieme di (al massimo) 12 bambini di 1-3 anni descritti da un'apposita classe `Bambino`. La classe dovrà prevedere metodi per eseguire almeno le seguenti operazioni: inserire un nuovo bambino (se possibile); rimuovere un bambino (per nome); visualizzare l'elenco dei bambini in ordine di età e, in subordine, alfabetico; rimuovere tutti i bambini di 3 anni e incrementare di 1 l'età di tutti gli altri. NOTA: si suggerisce di porre attenzione al metodo `compareTo` della classe `Bambino`.

Facoltativo: Includere nel programma un'interfaccia testuale con un menù che consenta di eseguire le funzionalità della classe `AsiloNido`.

Progetto 1.1.2 Realizzare una classe `CodaRistorante` per gestire l'arrivo dei clienti a un ristorante. Ogni gruppo di clienti è rappresentato da una classe `Clienti` che include almeno un nome e il numero delle persone che costituiscono il gruppo. La classe `CodaRistorante` dovrà prevedere metodi per eseguire almeno le seguenti operazioni: mettere in coda un nuovo gruppo di clienti; rimuovere dalla coda il primo gruppo di clienti di dimensione minore o pari a un valore passato come parametro (per rappresentare la situazione in cui si libera un tavolo di tale dimensione);

Facoltativo: Includere nel programma un'interfaccia testuale con un menù che consenta di eseguire le funzionalità della classe `CodaRistorante`.

Facoltativo: Utilizare la classe `CodaRistorante` all'interno di una classe `Ristorante` che include anche una lista di tavoli. Ogni tavolo avrà un certo numero di posti e sarà libero o occupato. La classe `Ristorante` prevederà metodi per descrivere almeno le seguenti operazioni: arrivo di un nuovo gruppo di clienti (si veda `CodaRistorante`); liberazione di un tavolo attualmente occupato.

2. Progetti che prevedono di approfondire in autonomia qualche classe del Java Collections Framework non vista a lezione (per max 3 persone)

Progetto 1.2.1 Rappresentare una rete sociale tramite una mappa (classe `HashMap` del Java Collections Framework) che associa stringhe (nomi di persone) a insiemi di stringhe (nomi di altre persone). Operazioni obbligatorie: inserire una nuova persona (un nuovo elemento alla mappa), aggiungere o togliere un'amicizia, visualizzare la rete sociale. Operazioni facoltative: visualizzare gli amici a distanza 2, rimuovere dall'insieme degli amici di una data persona tutte le persone che sono amiche di un'altra data persona, salvare/leggere la rete sociale da file di testo oppure usando la serializzazione di Java.

Progetto 1.2.2 Realizzare una classe `CentroAnalisi` per gestire gli appuntamenti (visite) di pazienti da parte dei medici della struttura. La classe gestisce gli appuntamenti di un singolo giorno, e deve garantire che le visite (che possono avere durata variabile) non si sovrappongano. Diversi medici sono disponibili, rappresentati da oggetti di tipo `Medico` che contengono almeno il nome del medico e la sua specializzazione (ad es. oculista, ortopedico, ecc..). I paziente sono rappresentati da una classe `Paziente` che ne descrive almeno il nome e il numero di telefono.

Ogni appuntamento deve essere memorizzato in una classe `Appuntamento`, che contiene almeno i riferimenti al medico e al paziente, l'orario di inizio e l'orario di fine della visita.

La lista degli appuntamenti deve essere memorizzata in una mappa (classe `HashMap` del Java Collections Framework) che associa al nome di ogni medico (una stringa) la lista dei suoi appuntamenti.

Si implementino metodi per eseguire le seguenti funzionalità: 1. richiedere un nuovo appuntamento, specificando ora di inizio, ora di fine, nome medico: se disponibile, deve essere registrata la prenotazione per il paziente corrispondente; 2. cercare la prima disponibilità del giorno per fissare un appuntamento di una certa durata fornita come parametro; 3. stampare la lista di tutti gli appuntamenti per un singolo medico; 4. stampare tutti gli appuntamenti di un singolo utente specificato tramite il nome; 5. eliminare un appuntamento di un utente, identificato specificando il nome dell'utente e l'ora di inizio dell'appuntamento.

Facolativo: Estendere il progetto prevedendo la disponibilità di un numero fissato di stanze (minore del numero di medici). Al fine di poter fissare un appuntamento con un medico deve essere disponibile una stanza in cui effettuare la visita.

3. Progetti che prevedono lo studio individuale di qualche libreria (o tecnologia) più avanzata (per max 5 persone)

Progetto 1.3.1 Scrivere un programma che gestisce un archivio di qualche tipo (una rubrica telefonica, un elenco di libri di una biblioteca, ecc...) memorizzato in un file in formato XML. Il programma dovrà consentire di effettuare operazioni di vario tipo sull'archivio (ad esempio: visualizzazione, inserimento/rimozione di elementi, filtraggio di elementi, ecc...) aggiornando di conseguenza il documento XML. Utilizzare il "Java DOM parser" della libreria di Java per generare un albero che rappresenta la struttura del documento XML.

Progetto 1.3.2. Realizzare un programma per gestire un torneo di calcio. Dapprima è necessario registrare la lista delle squadre, che deve essere inserita dall'utente oppure letta da un file. Quindi deve essere generato casualmente il calendario degli incontri, organizzato in due gironi (andata e ritorno). In ogni *giornata*, ogni squadra gioca una partita (a meno che non siano dispari...). Alla fine del torneo, ogni squadra deve aver incontrato ogni altra squadra esattamente due volte. Per la generazione del calendario degli incontri si veda l'algoritmo di Berger (`https://it.wikipedia.org/wiki/Algoritmo_di_Berger`) o algoritmi simili disponibili in rete;

L'esito di ogni partita deve essere generato casualmente, andando a determinare i gol segnati da ognuna delle due squadre.

Si implementino metodi per eseguire le seguenti funzionalità: 1. iniziare una nuova stagione, generando casualmente il tabellone degli incontri; 2. generare casualmente l'esito di ogni partita; 3. visualizzare i risultati di una giornata a scelta dell'utente; 4. visualizzare la classifica finale; 5. calcolare il numero di partite vinte/perse/pareggiate da ogni squadra, al termine della stagione;

Facoltativo: esportare i risultati delle singole giornate e la classifica finale in un documento HTML.