Esercitazione di laboratorio n. 1

Esercizio n. 1: Gestione Ordini in Java (livello base)

E' necessario scrivere un'applicazione Java che consente di effettuare la gestione degli ordini di una fabbrica di ricambi automobilistici. Gli ordini sono rappresentati da (a) un codice di 6 caratteri (es: "AB56XX"); (b) la descrizione dell'azienda che ha effettuato l'ordine (es: "Autofficina Rossi Srl") (c) l'elenco dei pezzi ordinati; per ciascun pezzo è necessario rappresentare (c.1) il codice numerico (es: 5647), (c.2) la descrizione (es: "carburatore Fiat Punto"), (c.3) la quantità (es: 1), (c.4) il costo unitario (es: 70 Euro). I dati di ciascun ordine sono memorizzati in documenti XML salvati sul disco. I documenti devono essere validi rispetto ad un DTD creato dallo sviluppatore. L'applicazione deve consentire di caricare, modificare e salvare dati relativi agli ordini. In particolare, deve essere possibile eseguire i seguenti casi d'uso

Caso d'Uso: "Utente carica ordine"

- Scenario principale
  - l'utente fornisce il nome di un file su disco
  - il sistema carica dal file i dati relativi all'ordine
  - il sistema visualizza i dati dell'ordine sullo schermo
- Scenari alterativi:
  - Errore nella lettura da file: l'applicazione segnala il problema all'utente stampando un messaggio e il caso d'uso si conclude
  - Ordine non valido rispetto al DTD: l'applicazione segnala il problema all'utente stampando un messaggio e il caso d'uso si conclude

Caso d'Uso: "Utente inserisce pezzo"

- Scenario principale
  - dato un ordine precedentemente caricato dal disco, l'utente fornisce i dati di un nuovo pezzo ordinato (codice, descrizione, quantità e costo unitario)
  - nel caso in cui l'ordine non contenga già un pezzo con lo stesso codice, il sistema aggiunge il pezzo all'ordine; nel caso in cui l'ordine contenga già un pezzo con lo stesso codice, modifica semplicemente la quantità ordinata.
  - il sistema visualizza i dati dell'ordine aggiornato sullo schermo

Caso d'Uso: "Utente salva ordine"

- Scenario principale
  - l'utente fornisce il nome di un file su disco
  - il sistema salva i dati dell'ordine nel file, crea una copia del relativo DTD
- Scenari alterativi:
  - Errore nella scrittura sa file: l'applicazione segnala il problema all'utente stampando un messaggio e il caso d'uso si conclude

Scrivere l'applicazione Java che effettua le operazioni elencate sopra, secondo le seguenti specifiche:

- Utilizzare l'architettura a tre strati vista a lezione (interfaccia&controllo, modello, persistenza)
- Sviluppare test di regressione utilizzando **JUnit** per le classi del modello (in particolare, per scrivere test su scrittura e caricamento, è opportuno predisporre dei file di dati di test e verificare che vengano letti e scritti correttamente)
- E' possibile utilizzare JDOM oppure JAXP come libreria per XML

Esercizio n. 2: Gestione Ordini in C# (livello intermedio)

Sviluppare l'applicazione descritta all'Esercizio n. 1 in C#.

Esercizio n. 3: Bowling in Java (livello avanzato)

E' necessario scrivere il sistema informativo di una sala bowling che consenta la gestione dei tabelloni elettronici segnapunti collocati sulle varie piste. In particolare, il sistema deve tenere traccia dei punteggi ottenuti dai giocatori.
Le regole del bowling prevedono che ogni partita si gioca tra due giocatori (o due squadre). Ciascun giocatore gioca 10 "**frame**"; in ogni frame l'obiettivo è buttare giù i 10 birilli sulla pista. Il giocatore ha a disposizione due palle: a seconda dei birilli abbattuti con le due palle si ottengono punteggi diversi:

- I caso: birilli rimanenti: se il giocatore non abbatte tutti i birilli utilizzando i due tiri, il suo punteggio è pari ai birilli abbattuti
- II caso: "**spare**": il giocatore abbatte tutti i birilli con la seconda palla (es: 3 + 7 birilli abbattuti); si tratta di uno "spare"; in questo caso il suo punteggio è pari ai birilli abbattuti (10), più i birilli abbattuti con la palla successiva; ovvero la prima palla del prossimo frame o un tiro aggiuntivo se siamo nel 10 frame
- III caso: "**strike**": il giocatore abbatte tutti i birilli con la prima palla: si tratta di uno "strike"; il frame si conclude con la prima palla; il suo punteggio è pari a 10 più i birilli abbattuti con le due palle successive; le due palle del prossimo frame o il prossimo strike più la palla successiva o tiri aggiuntivi se siamo nel 10 frame

I punteggi riportati da ciascun giocatore vengono stampati al termine della partita su una "**score card**" in cui vengono riassunti i punteggi di ciascun frame. Di seguito vengono riportati alcuni esempi di score card (nelle score card ciascun riquadro rappresenta un frame; nel riquadro sono riportati in alto i punteggi ottenuti con i tiri, in basso il punteggio dopo quel frame; il carattere "/" rappresenta uno spare, il carattere "X" uno strike):

1 4	4 5	6 /	5 /	X	0 1	7 /	6 /	X	2 / 6
5	14	29	49	60	61	77	97	117	133

2 1	3 /	X	2 2	2 0	X	X	X	1 /	X 2 /
3	23	37	41	43	73	94	114	134	154

X	X	X	X	X	X	X	X	X	X X X
30	60	90	120	150	180	210	240	270	300

L'ultima score card rappresenta un "gioco perfetto", ovvero una sequenza di 12 strike, che ottiene il punteggio massimo di 300 punti.
L'applicazione deve consentire di sviluppare i seguenti casi d'uso:

Caso d'Uso: "Giocatori giocano partita"

- Scenario principale
  - due giocatori a cui è stata assegnata una pista iniziano la partita fornendo il loro nome
  - a turno effettuano i tiri; il sistema meccanico della pista rileva i birilli abbattuti e aggiorna il punteggio di ciascun giocatore visualizzandolo sullo schermo
  - al termine della partita il sistema decreta il vincitore e stampa le due score card dei giocatori

---

Istruzioni per la creazione dell'applicazione

- Effettuare il "login" sulla macchina assegnata utilizzando come note utente la propria matricola e come password la password assegnata dal CISIT
- Creare i file della propria applicazione nella cartella "**Documenti**" del disco C: (**c:\Documents and Settings\<matricola>\Documenti**); es: lo studente di matricola 12654 lavorerà nella cartella c:\Documents and Settings\12654\Documenti
- Per eseguire il compilatore, lanciare il prompt dei comandi (*Start >> Tutti i Programmi >> Accessori >> Prompt dei Comandi*), e spostarsi nella cartella **Documenti** con le seguenti istruzioni:
  - `cd \Documents*`
  - `cd <matricola> (es: cd 12654)`
  - `cd Documenti`
- La documentazione del linguaggio Java è disponibile nella cartella **c:\j2sdk1.4.2\docs**. Per consultarla aprire il file **index.html** con un qualsiasi browser Web
- La documentazione del framework .NET è disponibile a partire dal file **startHere.htm** nella cartella **C:\Programmi\Microsoft.NET\SDK\v1.1**; aprire il file utilizzanto Internet Explorer
- **JUnit** è già installato sulle macchine ed aggiunto al CLASSPATH; per eseguire il testRunner grafico è possibile utilizzare il comando **java junit.swingui.TestRunner** <nomeDellaClasseDiTest>
- La classe **it.unibas.utilita.Console** è disponibile nella cartella **c:\lib**
- Durante l'utilizzo di una macchina sarà disponibile – come sui calcolatori del Centro di Calcolo Studenti del CISIT – la propria cartella personale di rete, montata come disco Z:; è possibile salvare il lavoro svolto sul disco Z: in modo da poterlo recuperare successivamente anche dalle altre postazioni di lavoro
- Al termine dell'esercitazione disconnettersi e NON spegnere il computer