PROGRAMMAZIONE II - Anno Accademico 2014-15 - Progetto Sessione Estiva

Il progetto ha l'obiettivo di applicare i concetti e le tecniche di programmazione esaminate durante il corso: consiste nella realizzazione di alcuni moduli software, ed è strutturato in due esercizi di programmazione.

Esercizio 1: Progettazione e sviluppo di un interprete in OCaml

Si consideri un semplice linguaggio funzionale la cui sintassi concreta è definita dalla seguente grammatica

(Identificatori)	Ide ::= <definizione standard>	
(Interi)	Int ::= <definizione standard>	
(Valori)	Val :: = Int	(Interi)
	| True | False	(Valori Booleani)
	| fun Ide -> E	(Funzioni con un solo parametro, non ricorsive)
(Espressioni)	E ::= Ide	(Identificatori)
	| Val	(Valori)
	| E and E | E or E | not E	(Espressioni booleane)
	| OP(E,E)	(Espressioni su interi con OP ∈{"+", "-", "*", "=", "<="})
	| if E then E else E	(Condizionale)
	| let Ide = E in E	(Blocco let)
	| E(E)	(Applicazione funzionale)
	| try Ide with E in P	(Espressione try ... with ...)
(Pattern)	P ::= (_ --> E)	(Default Pattern)
	| (E --> E)	(Pattern elementare)
	| (E --> E) :: P	(Composizione di pattern)

Il significato dei costrutti del linguaggio è standard, tranne l'espressione **try** x ... **with** …, che è utilizzata per ottenere una forma di pattern matching. Nella valutazione di **try** x **with** E **in** P, il valore ottenuto dalla valutazione dell'espressione E è legato alla variabile x e determina la selezione del pattern con la conseguente esecuzione dell'espressione associata. L'idea intuitiva è che la prima espressione del pattern contiene la variabile x e che l'espressione opera come una guardia booleana: il pattern è selezionato se la guardia è verificata, e in tal caso viene eseguita l'espressione associata. Il pattern (_ --> E) è di default.

Ad esempio, l'espressione seguente associa alla variabile x il valore della valutazione di +(z,w) ed esegue il pattern selezionato dalla verifica della condizione, a seconda che +(z,w) sia maggiore, minore o uguale a 0.

```
try x with +(z,w) in
  (x > 0 --> succ(x)) ::
  (x < 0 --> abs(x)) ::
  (_ --> x)
```

1. Si definisca una sintassi astratta per il linguaggio, introducendo opportuni tipi di dati OCaml.
2. Si definisca in OCaml un interprete del linguaggio assumendo la regola di scoping statico.
3. Si verifichi la correttezza dell'interprete progettando ed eseguendo una quantità di casi di test sufficiente a verificare tutti gli operatori.
4. Si descriva quali sono le modifiche da apportare all'interprete del linguaggio nel caso si voglia assumere una regola di scoping dinamico.

Esercizio 2: Progettazione e realizzazione di un modulo Java

Lo scopo del progetto è lo sviluppo di una componente software di supporto alla gestione di un semplice sistema di microblogging. Da un punto di vista astratto un microblog può essere visto come una collezione di contenuti di testo con una lunghezza massima. Per semplicità assumiamo che i contenuti siano testuali e siano rappresentati da stringhe lunghe al massimo di 140 caratteri. Assumiamo, inoltre, che il microblog sia utilizzabile solamente tramite una registrazione: solo gli utenti registrati possono scrivere e leggere i messaggi presenti nel microblog.

La struttura astratta del microblog è caratterizzata dalla interfaccia Java definita di seguito.

```
interface SimpleTw {
void addUser(User bob, String pass) throws UnauthorizedAccessException;
        // Aggiunge l'utente bob all'insieme degli utenti. Solleva una eccezione se pass non è corretta.
void deleteUser(User bob, String pass) throws UnauthorizedAccessException;
        // Elimina l'utente bob dall'insieme degli utenti. Solleva una eccezione se pass non è corretta.
int tweet(String message, Tag t, User bob)
                    throws UnauthorizedUserException, MsgException;
        // Inserisce un messaggio di bob con intestazione t. Restituisce un codice numerico del messaggio.
        // Lancia un'eccezione se bob non è un utente registrato o se il messaggio supera la dimensione massima.
String readLast(User bob) throws EmptyMsgException;
        // Restituisce l'ultimo messaggio inserito da bob. Solleva un'eccezione se non ci sono messaggi di bob
String readLast() throws EmptyMsgException;
        // Restituisce l'ultimo messaggio inserito. Solleva un'eccezione se non sono presenti messaggi.
List<String> readAll(Tag t);
        // Restituisce tutti i messaggi inseriti con Tag t, nell'ordine di inserimento.
List<String> readAll();
        // Restituisce tutti i messaggi inseriti, nell'ordine di inserimento.
void delete(int code) throws WrongCodeException;
        // Cancella il messaggio identificato da code.
        // Solleva un'eccezione se non esiste un messaggio con quel codice.
boolean empty();
        // Restituisce true se e solo se non sono presenti messaggi.
}
```

1. Si definisca la classe **User** (con un'unica variabile d'istanza **String nick**) e le classi di eccezioni usate nell'interfaccia, in modo da compilare l'interfaccia con successo. Il tipo generico **List<E>** è quello fornito dalla libreria standard di Java.
2. Si definisca un'implementazione dell'interfaccia **SimpleTw**, chiamata **myTw**. Il costruttore di **myTw** ha un argomento di tipo **String** che serve per fornire una password (non modificabile) quando si crea un oggetto della classe. Non sono posti vincoli sulla struttura di implementazione prescelta.
3. Si descriva la funzione di astrazione e l'invariante di rappresentazione. Si realizzi una batteria di test per valutare il comportamento dell'implementazione proposta.

Modalità di consegna Progetto Sessione Estiva

- Il progetto deve essere svolto e discusso col docente individualmente. Il confronto con colleghi mirante a valutare soluzioni alternative durante la fase di progetto è incoraggiato.
- Per poter sostenere l'orale di un appello, il progetto deve essere consegnato entro il giorno dello scritto dello stesso appello, quindi entro (le ore 24 di) mercoledì 10 giugno per il primo appello, e entro martedì 7 luglio per il secondo appello. Un progetto sottomesso per il primo appello vale anche per il secondo se lo studente non supera la prova scritta o quella orale. Il progetto vale comunque solo per la sessione estiva.
- Il progetto deve essere costituito da
    - i file sorgente contenenti il codice sviluppato e le corrispondenti batterie di test, ove tutto il codice deve essere adeguatamente commentato;
    - una relazione di massimo una pagina per esercizio, che descrive le principali scelte progettuali ed eventuali istruzioni per eseguire il codice.
- La consegna va fatta inviando per email tutti i file in un archivio. Per il corso A, inviare l'email al Prof. Ferrari con oggetto "[PR2A] Consegna progetto". Per il corso B, inviare l'email al Prof. Gadducci con oggetto contenente la stringa "[PR2B] Consegna progetto".

Altre informazioni

- Per quanto riguarda il progetto, i docenti risponderanno solo a eventuali domande riguardanti l'interpretazione del testo, e non commenteranno soluzioni parziali prima della consegna.