Appelli di gennaio, febbraio e aprile 2006

MyLanguage

Laurea triennale in Comunicazione Digitale
Laboratorio di Programmazione

1 – Descrizione

Il progetto consiste nell'implementare in JAVA un semplice linguaggio di programmazione, in cui le variabili sono esclusivamente di tipo intero e vengono azzerate automaticamente durante la loro dichiarazione.

2 – Le classi da realizzare

E' richiesto di risolvere il progetto descritto nella sezione precedente realizzando in JAVA le seguenti classi:

- `Istruzione`, che descrive una generica istruzione del linguaggio. La classe dovrà contenere:
  - la definizione della Hashtable `memoria`, deputata a contenere i valori memorizzati nelle variabli, utilizzando come chiave di ricera i nomi dati alle variabili; attenzione: la definizione di questa hashtable dovrà essere compatibile con il resto delle definizioni fatte nella classe, e, soprattutto, dovrà essere fatta in modo da essere leggibile e scrivibile da tutte le istanze delle classi derivate da `Istruzione`, nonché condivisa da queste istanze;
  - la definizione del metodo astratto `void esegui()`, da chiamare per eseguire l'istruzione; durante l'esecuzione di tale metodo potrà essere lanciata l'eccezione `ExecutionException`, rappresentante un generico errore di run time;
  - la definizione del metodo astratto `Object clone()`, da chiamare per restituire una **copia** dell'istruzione corrente;
  - la definizione del metodo astratto `String toString()`, da chiamare per ritornare una descrizione testuale dell'istruzione; attenzione: tale descrizione dovrà essere opportunamente indentata nel caso venga stampata un'istruzione contenente una delle strutture `Sequenza`, `Selezione` o `Iterazione` descritte in seguito;
- `Sequenza`, che descrive una generica sequenza di istruzioni. La classe dovrà contenere:
  - la variabile di istanza `Istruzione elenco[]`, deputata a contenere l'insieme delle istruzioni da eseguire in sequenza;

- il costruttore `public Sequenza(Istruzione e[])`, che inizializza la variabile di istanza della classe **copiando** in essa i contenuti dell'array passato come argomento;
- l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da eseguire in sequenza tutte le istruzioni contenute nell'array elenco, dalla prima all'ultima posizione;
- l'implementazione del metodo `toString` della superclasse dovrà essere fatta in modo da concatenare la descrizione testuale di tutte le istruzioni contenute in `elenco` (ottenibile chiamando il corrispondente metodo `toString`) nell'ordine in cui verranno eseguite;

- `Selezione`, che descrive l'esecuzione condizionata di un'istruzione sulla base dei contenuti di una variabile. La classe dovrà contenere:
  - le variabili di istanza `Istruzione istrSe` e `Istruzione istrAltrimenti`, che identificano due istruzioni;
  - la variabile di istanza `String variabile`, che contiene il nome di una variabile;
  - la variabile di istanza `String operazione`, che dovrà contenere la descrizione dell'operazione di confronto da utilizzare; i valori possibili per questa variabile sono `">"`, `">="`, `"<"`, `"<="`, `"=="` e `"!="`, il cui significato è mutuato da quello degli analoghi operatori in JAVA;
  - la variabile di istanza `int valore`, che dovrà contenere il valore da confrontare con il contenuto della variabile;
  - il costruttore `public Selezione(Istruzione se, Istruzione altrimenti, String var, String o, int v)`, che inizializza le variabili di istanza della classe **copiando** in essa le istruzioni, le stringhe e la variabile intera passate come argomento, tenendo conto del fatto che l'ordine degli argomenti equivale all'ordine in cui sono state presentate le variabili di istanza;
  - l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da valutare una condizione booleana confrontando, utilizzando l'operatore descritto in `operazione`, il valore contenuto nella variabile il cui nome è contenuto in `variabile` con il contenuto di `valore`. Quando questa condizione è vera, verrà eseguita l'istruzione identificata da `istrSe`, mentre quando la condizione è falsa verrà eseguita `istrAltrimenti`; il metodo dovrà lanciare l'eccezione `InvalidVariableException` qualora non sia stata precedentemente dichiarata (attraverso l'istruzione `Dichiara`, descritta in seguito) una variabile il cui nome è contenuto in `variabile`; dovrà inoltre essere lanciata l'eccezione `InvalidOperandException` nel caso in cui la descrizione dell'operazione non contenga uno dei simboli `">"`, `">="`, `"<"`, `"<="`, `"=="` o `"!="`;
  - l'implementazione del metodo `toString` dovrà essere fatta in modo da ritornare una stringa che contenga, su linee differenti:
    - la stringa `"IF"`, un carattere di aperta parentesi tonda, il nome della variabile su cui agisce l'istruzione, la stringa che descrive

l'operatore di confronto utilizzato (`">"`, `">="`, `"<"`, `"<="`, `"=="` o `"!="`), il valore rispetto a cui confrontare la variabile e un carattere di chiusa parentesi tonda,
    - la descrizione testuale dell'istruzione identificata da `istrSe`, **indentata spostandosi di due spazi in avanti** rispetto alla linea precedente,
    - la stringa "ELSE", **indentata spostandosi di due spazi indietro** rispetto alla linea precedente
    - la descrizione testuale dell'istruzione identificata da `istrAltrimenti`, **indentata spostandosi di due spazi in avanti** rispetto alla linea precedente;
- `Iterazione`, che descrive l'esecuzione iterata di un'istruzione sulla base del contenuto di una variabile. La classe dovrà contenere:
    - la variabile di istanza `Istruzione corpo`, che identifica l'istruzione da eseguire;
    - la variabile di istanza `String variabile`, che contiene il nome di una variabile;
    - la variabile di istanza `String operazione`, che dovrà contenere la descrizione dell'operazione di confronto da utilizzare; i valori possibili per questa variabile sono `">"`, `">="`, `"<"`, `"<="`, `"=="` e `"!="`, il cui significato è mutuato dal significato degli analoghi operatori in JAVA;
    - la variabile di istanza `int valore`, che dovrà contenere il valore da confrontare con il contenuto della variabile;
    - il costruttore `public Iterazione(Istruzione c, String v, String o, int val)`, che inizializza le variabili di istanza della classe **copiando** in esse la stringa contenente il nome della variabile, l'istruzione, la stringa che identifica l'operatore di confronto e la variabile intera passate come argomento, tenendo conto del fatto che l'ordine degli argomenti equivale all'ordine in cui sono state presentate le variabili di istanza;
    - l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da eseguire l'istruzione identificata da `corpo` fintantoché confrontando il valore contenuto nella variabile identificata da `variabile` con il valore contenuto in `valore` utilizzando l'operatore indicato da `operatore` si ottenga un'asserzione logicamente vera (in analogia con il ciclo while del linguaggio JAVA); il metodo dovrà lanciare l'eccezione `InvalidVariableException` qualora non sia stata precedentemente dichiarata (attraverso l'istruzione `Dichiara`, descritta in seguito) una variabile il cui nome è contenuto in `variabile`; dovrà inoltre essere lanciata l'eccezione `InvalidOperandException` nel caso in cui la descrizione dell'operazione non contenga uno dei simboli `">"`, `">="`, `"<"`, `"<="`, `"=="` o `"!="`;
    - l'implementazione del metodo `toString` dovrà essere fatta in modo da ritornare una stringa contenente, su linee differenti:

        - la stringa `"WHILE"`, un carattere di aperta parentesi tonda, il nome della variabile su cui agisce l'istruzione, la stringa che descrive l'operatore di confronto utilizzato (`">"`, `">="`, `"<"`, `"<="`, `"=="` o `"!="`), il valore rispetto a cui confrontare la variabile e un carattere di chiusa parentesi tonda,
        - la descrizione testuale dell'istruzione identificata da `corpo`, **indentata spostandosi di due spazi in avanti** rispetto alla linea precedente;
- `Dichiara`, che descrive la dichiarazione di una variabile. La classe dovrà contenere:
    - la variabile di istanza `String variabile`, che contiene il nome di una variabile;
    - il costruttore `public Dichiara(String v)`, che inizializza la variabile di istanza della classe copiando in essa il valore passato come argomento;
    - l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da inserire nella hashtable della superclasse una variabile avente nome uguale a quello contenuto in `variabile` e inizializzata a 0; il metodo dovrà lanciare l'eccezione `ExistingVariableException` qualora l'argomento `variabile` identifichi una variabile precedentemente dichiarata;
    - l'implementazione del metodo `toString` dovrà essere fatta in modo da ritornare una stringa contenente la stringa "DECLARE" seguita da un carattere di spazio, dal nome della variabile, da un carattere di punto e virgola e da un carattere di newline (a capo);
- `Assegna`, che descrive l'esecuzione di assegnamento di un valore ad una variabile. La classe dovrà contenere:
    - la variabile di istanza `String variabile`, che identifica il nome di una variabile;
    - la variabile di istanza `Integer valore`, che identifica il valore intero da assengare alla variabile (codificato come istanza della classe `Integer`);
    - il costruttore `public Assegna(String var, int val)`, che inizializza le variabili di istanza della classe copiando in esse i valori passati come argomento;
    - l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da assegnare il valore contenuto nella variabile `valore` alla variabile il cui nome è identificato da `variabile`, cancellandone il contenuto precedente; il metodo dovrà lanciare l'eccezione `InvalidVariableException` qualora l'argomento `variabile` non identifichi una variabile precedentemente dichiarata;
    - l'implementazione del metodo `toString` dovrà essere fatta in modo da ritornare una stringa contenente il nome della variabile seguito da uno spazio, un carattere "=", un altro spazio, il contenuto di `valore`, un carattere di punto e virgola e un carattere di newline (a capo);

- `Incrementa`, che descrive l'incremento del valore memorizzato in una variabile di una preassegnata quantità. La classe dovrà contenere:
  - la variabile di istanza `String variabile`, che identifica la variabile da incrementare;
  - la variabile di istanza `int incremento`, che indica di quanto incrementare il contenuto della variabile;
  - il costruttore `public Incrementa(String v, int incr)`, che inizializza le variabili di istanza della classe copiando in esse i valori passati come argomento;
  - l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da aggiungere il contenuto della variabile `incremento` al valore contenuto nella variabile identificata da `variabile`; il metodo dovrà lanciare l'eccezione `InvalidVariableException` qualora la variabile non sia stata precedentemente dichiarata e l'eccezione `InvalidIncrementException` qualora `incremento` non contenga un valore strettamente positivo;
  - l'implementazione del metodo `toString` dovrà essere fatta in modo da ritornare una stringa contenente il nome della variabile seguito da un carattere di spazio, dalla stringa "+=", da un altro carattere di spazio, dal contenuto di `incremento`, da un carattere di punto e virgola e da un carattere di newline (a capo);
- `Decrementa`, che descrive il decremento del valore memorizzato in una variabile di una preassegnata quantità. La classe dovrà contenere:
  - la variabile di istanza `String variabile`, che identifica la variabile da decrementare;
  - la variabile di istanza `int decremento`, che indica di quanto decrementare il contenuto della variabile;
  - il costruttore `public Decrementa(String v, int decr)`, che inizializza le variabili di istanza della classe copiando in esse i valori passati come argomento;
  - l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da sottrarre il contenuto della variabile `decremento` al valore contenuto nella variabile identificata da `variabile`; il metodo dovrà lanciare l'eccezione `InvalidVariableException` qualora la variabile non sia stata precedentemente dichiarata e l'eccezione `InvalidDecrementException` qualora `decremento` non contenga un valore strettamente positivo;
  - l'implementazione del metodo `toString` dovrà essere fatta in modo da ritornare una stringa contenente il nome della variabile seguito da un carattere di spazio, dalla stringa "-=", da un altro carattere di spazio, dal contenuto di `incremento`, da un carattere di punto e virgola e da un carattere di newline (a capo);
- `StampaVariabile`, che descrive l'output a video del valore memorizzato in una variabile. La classe dovrà contenere:
  - la variabile di istanza `String variabile`, che identifica la variabile di cui stampare il contenuto;

- il costruttore `public StampaVariabile(String v)`, che inizializza la variabile di istanza della classe copiando in essa il valore passato come argomento;
- l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da stampare a video il valore contenuto nella variabile identificata da `variabile`; il metodo dovrà lanciare l'eccezione `InvalidVariableException` qualora `variabile` non identifichi una variabile precedentemente dichiarata;
- l'implementazione del metodo `toString` dovrà essere fatta in modo da ritornare una stringa contenente la stringa "PRINT" seguita da un carattere di spazio, dal nome della variabile da stampare, da un carattere di punto e virgola e da un carattere di newline (a capo);

- `Stampa`, che descrive l'output a video di un generico messaggio. La classe dovrà contenere:
  - la variabile di istanza `String messaggio`, che identifica il messaggio da visualizzare;
  - il costruttore `public Stampa(String msg)`, che inizializza la variabile di istanza della classe **copiando** in essa il valore passato come argomento;
  - l'implementazione del metodo `esegui` della superclasse dovrà essere fatta in modo da stampare a video la stringa contenuta nella variabile di istanza;
  - l'implementazione del metodo `toString` dovrà essere fatta in modo da ritornare una stringa contenente la stringa "PRINT" seguita da un carattere di spazio, dal messaggio da visualizzare (delimitato da doppi apici), da un carattere di punto e virgola e da un carattere di newline (a capo);
- Le classi `ExecutionException`, `InvalidVariableException`, `ExistingVariableException`, `InvalidIncrementException`, `InvalidDecrementException` e `InvalidOperandException` definite in modo opportuno;

A parte quanto espressamente richiesto, è lasciata piena libertà sull'implementazione delle singole classi e sull'eventuale introduzione di classi aggiuntive, a patto di seguire le regole del paradigma ad oggetti ed i principi di buona programmazione. Si suggerisce di porre particolare attenzione alla scelta dei modificatori relativi a variabili d'istanza e metodi, nonchè alla dichiarazione e alla gestione delle eccezioni che possono venire lanciate dai vari metodi e alle relazioni di ereditarietà tra le varie classi.

Si suggerisce altresì di porre particolare cura all'implementazione dei metodi `esegui` e `toString`, tenendo conto anche di casi in cui l'istruzione da eseguire o da visualizzare contenga in realtà una struttura complessa, costruita utilizzando più volte `Sequenza`, `Selezione` ed `Iterazione`.

Non è richiesto l'utilizzo di particolari modalità grafiche di visualizzazione: è sufficiente una qualunque modalità di visualizzazione basata sull'uso dei caratteri.

E' invece **espressamente richiesto** di non utilizzare package non standard di Java (si possono quindi utilizzare `java.util`, `java.io` e così via), con l'unica eccezione package `prog.io` incluso nel libro di testo per gestire l'input da tastiera e l'output a video.

3 – Programma.java

Non verranno presi in considerazione progetti le cui classi non permetteranno almeno di compilare il seguente sorgente:

```
class Programma {
  public static void main(String args[]) {
    try {
      Dichiara d0 = new Dichiara("s");
      int argom = Integer.parseInt(args[0]);
      Dichiara d1 = new Dichiara("arg1");
      Assegna a1 = new Assegna("arg1", argom);
      Istruzione[] id = new Istruzione[2];
      id[0] = new Decrementa("arg1", 1);
      id[1] = new Incrementa("s", 1);
      Sequenza s = new Sequenza(id);
      Iterazione w1 = new Iterazione("arg1", s, "!=", 0);
      argom = Integer.parseInt(args[1]);
      Dichiara d2 = new Dichiara("arg2");
      Assegna a2 = new Assegna("arg2", argom);
      id[0] = new Decrementa("arg2", 1);
      id[1] = new Incrementa("s", 1);
      s = new Sequenza(id);
      Iterazione w2 = new Iterazione("arg2", s, "!=", 0);
      StampaVariabile pr = new StampaVariabile("s");
      Istruzione[] p = new Istruzione[3];
      p[0] = w1;
      p[1] = w2;
      p[2] = pr;
      Sequenza s1 = new Sequenza(p);
      Stampa err2 = new Stampa("Secondo argomento invalido");
      Selezione chk2 = new Selezione(err2, s1, "arg2", "<=", 0);
      id = new Istruzione[3];
      id[0] = d2;
      id[1] = a2;
      id[2] = chk2;
      Sequenza s2 = new Sequenza(id);
      Stampa err1 = new Stampa("Primo argomento invalido");
      Selezione chk1 = new Selezione(err1, s2, "arg1", "<=", 0);
      id = new Istruzione[4];
      id[0] = d0;
      id[1] = d1;
      id[2] = a1;
      id[3] = chk1;
      Sequenza s3 = new Sequenza(id);
      System.out.println(s3);
      s3.esegui();
    } catch(ExecutionException e) {
```

```
            e.printStackTrace();
        }
    }
}
```

Naturalmente la discussione del progetto verterà **anche** sull'analisi dell'algoritmo codificato in Programma.java e sul peculiare uso delle variabili in esso contenute.

4 – Modalità di consegna

Il progetto può essere svolto al massimo da tre persone che intendono sostenere l'intero esame di Fondamenti di Architettura e Programmazione - Laboratorio di Programmazione negli appelli di Gennario, Febbraio o Aprile 2006, e deve essere consegnato **entro mezzanotte di Lunedì 13 febbraio 2006**, utilizzando il sito di sottoposizione delle esercitazioni (all'indirizzo `http://fap.dsi.unimi.it`). Per poter effettuare la sottoposizione è necessario autenticarsi utilizzando un nome di login e una password. Nella pagina principale del sito stesso è spiegato come ottenere questi dati. Nel caso il progetto venga svolto da più di una persona, dovrà essere fatta in ogni caso **una sola sottoposizione**, indicando chiaramente in un commento all'inizio dei sorgenti consegnati nome, cognome e matricola dei vari componenti del gruppo.

Dovranno essere consegnati tutti i **sorgenti** Java che permettano al programma di essere compilato ed eseguito correttamente

- compressi in un archivio di tipo ZIP che estragga i file nella directory in cui si trova l'archivio stesso, **oppure**
- contenuti in un unico file in cui tutte le classi **non devono essere dichiarate di tipo public**;

altri tipi di sottoposizioni verranno automaticamente rifiutate dal sito.

Il sistema rifiuterà automaticamente le sottoposizioni i cui sorgenti contengano errori rilevati in fase di compilazione.

E' inoltre richiesto di consegnare, **entro martedì 14 febbraio 2005**, una copia cartacea della stampa del codice sorgente prodotto in portineria del DSI o nella casella di posta del docente, indicando chiaramente nome, cognome e numero di matricola di tutti i componenti del gruppo, nonché il turno e il docente di riferimento, unitamente a un **breve** documento che descriva il modo in cui interfacciarsi con il programma e illustri le principali scelte implementative e le strategie utilizzate per svolgere il progetto. Nel caso si voglia consegnare questo documento **anche** tramite il sito di sottoposizione, sarà necessario utilizzare **un formato non proprietario** (**sono accettabili pdf, rtf e txt**).

6 – Valutazione

Durante la prova orale con i singoli studenti saranno discusse le modalità implementative adottate e la padronanza di alcuni dei concetti necessari per preparare il progetto e/o spiegati a lezione. La valutazione del progetto sarà fatta in base alla

- conformità dell'implementazione scelta per risolvere il problema con il paradigma di programmazione a oggetti;
- conformità del codice presentato alle regole di buona programmazione;
- adeguatezza del manuale utente presentato a descrivere il modo in cui un utente può utilizzare il programma;
- assenza di errori nel programma;
- usabilità del programma;

Dario Malchiodi
Dipartimento di Scienze dell'Informazione
Via Comelico 39/41 20135 Milano
Stanza T304 – Tel. +39 02 503 16338
eMail malchiodi@dsi.unimi.it

Walter Cazzola
Dipartimento di Informatica e Comunicazione
Via Comelico 39/41 20135 Milano
Stanza S233 – Tel. +39 0103536637
eMail cazzola@dico.unimi.it