Metodologie di Programmazione 2003 – 2004
TERZO APPELLO: 8 GIUGNO 2004

Nome: _____ Matricola: _____

Istruzioni

- Scrivete il vostro nome su tutti i fogli.
- Scrivete le soluzioni nello spazio riservato a ciascun esercizio.
- Giustificate le risposte: le risposte senza giustificazione non saranno considerate.
- Tempo a disposizione 2 ore e 30.
- No libri, appunti o altro.

LASCIATE IN BIANCO:

A1	
A2	
A3	
B1	
B2	
B3	
Totale	

Nome: _____ Matricola: _____

Nome: _____ Matricola: _____

Esercizio A1 Considerate la seguente gerarchia di classi:

```
interface M { M m(); }
interface N { void n(); }

class A implements M {
   public M m() { return new B(); }
}
class B extends A {
   public M m() { return new A(); }
}
class C extends A implements N {
   public void n() {}
}
```

Quale è il risultato della compilazione e della (eventuale, nel caso la compilazione non dia errori) esecuzione dei seguenti frammenti?

1. `N x = new C(); M y = x.m();`

2. `M x = new A(); B y = (B)x.m();`

3. `M x = new B(); B y = (B)x.m();`

Nome: _____ Matricola: _____

Nome: _____ Matricola: _____

Esercizio A2 Considerate le seguenti classi.

```
class A {
  void test(int d)
  { System.out.print("A"); }
}

class B extends A {
  void test(int i)
  { System.out.print("B-int"); }
  void test(char c)
  { System.out.print("B-char"); }
}
```

1. Cosa stampa `A a = new B(); a.test(1);`? Perchè?

2. Cosa stampa `A a = new B(); a.test('x');` ? Perchè?

3. Cosa stampa `B a = new B(); a.test('x');` ? Perchè?

Nome: _____ Matricola: _____

Nome: _____ Matricola: _____

Esercizio A3 Data la seguente definizione:

```
interface T {  boolean m();  }

class A implements T {
   boolean m() { return false; }
}

class B implements T {
   boolean m() { return true; }
}
```

definite il corpo del seguente metodo:

```
int countAs(List alist) {
  // scorre alist e restituisce il numero di elementi
  // che hanno tipo A









}
```

Nome: _____ Matricola: _____

Nome: _____ Matricola: _____

Parte B In questo set di esercizi dovete costruire un sistema di classi per simulare l'interprete di una semplice calcolatrice elettronica (CE) per valori reali. Ogni CE è dotata di

- un'area memoria, detta RAM, in cui memorizza i valori;
- un'area di memoria, detta PROG, in cui memorizza la sequenza di istruzioni che formano i programmi;
- quattro registri: A = accumula i risultati; IP = contiene l'indice dell'istruzione corrente; IR = contiene l'istruzione corrente; S = stato della macchina (stop o meno).

L'*instruction set* di una CE comprende le istruzioni nella tabella seguente.

Formato Simbolico	**Significato**
LOAD ind	A ← RAM[ind]
LOADC val	A ← val
STORE ind	RAM[ind] ← A
ADD ind	A ← A + RAM[ind]
MUL ind	A ← A * RAM[ind]
JUMP ind	IP ← ind
INCR	A ← A + 1
DECR	A ← A - 1
ALT	ferma l'esecuzione

Il ciclo di esecuzione è descritto dallo pseudo-codice seguente

```
procedure interpreta (start:integer)
begin
  IP ← start; S ← false
  while (not S)
    IR ← PROG[IP]
    if IR = (LOAD ind) then A ← RAM[ind]
    elsif IR = (STORE ind) then RAM[ind] ← A
    elsif ...
    ...
    elsif IR = ALT then S ← true
    endif
    if IR != (JUMP ind) then IP ← IP+1
end
```

Nome: _____ Matricola: _____

Nome: _____ Matricola: _____

Esercizio B1

Definite una classe `CE` che rappresenti una calcolatrice con le caratteristiche descritte in precedenza. La classe definisce

- opportuni campi per rappresentare le componenti della calcolatrice, e (uno o più) costruttori per inizializzare tali campi;
- un metodo `interpreta(int start, boolean trace)` che simula il comportamento della funzione 'interpreta' definita in precedenza ed inoltre, se `trace == true`, stampa ogni istruzione dopo averla eseguita.

Nome: _____ Matricola: _____

Nome: _____ Matricola: _____

Esercizio B2

Definite una classe astratta `Istruzione` che definisce un campo `String nome` e fornisce

- un costruttore che inizializza il campo nome al valore del parametro
- un metodo `void esegui(CE c)`, dove `CE` è la classe che rappresenta una calcolatrice
- un metodo `String toString()` che restituisce il valore del campo `nome`.

Definite inoltre tre sottoclassi `LOADC`, `INCR`, `ALT` della classe `Istruzione`, per rappresentare le corrispondenti istruzioni di una CE. Ognuna delle sottoclassi definisce almeno:

- un costruttore senza parametri che inizializza il campo `nome` alla stringa che corrisponde al nome della classe (ad esempio: nella classe `LOADC` il campo viene inizializzato alla stringa `"LOADC"`).
- il metodo `esegui(CE c)` che realizza il significato dell'operazione che corrisponde alla classe come indicato nella tabella a pag 4.

Nome: _____ Matricola: _____

Nome: _____ Matricola: _____

Esercizio B3

Definite una classe `Test` che testa la vostra implementazione creando una `CE` e facendogli eseguire la rappresentazione del seguente programma:

LOADI 0 INCR INCR ALT

e stampando la sequenza di istruzioni via via che le esegue.

Nome: _____ Matricola: _____