Corso di laurea in Comunicazione digitale — Fondamenti di architettura e programmazione

AA 2008/09 — Prof. M.A. Alberti

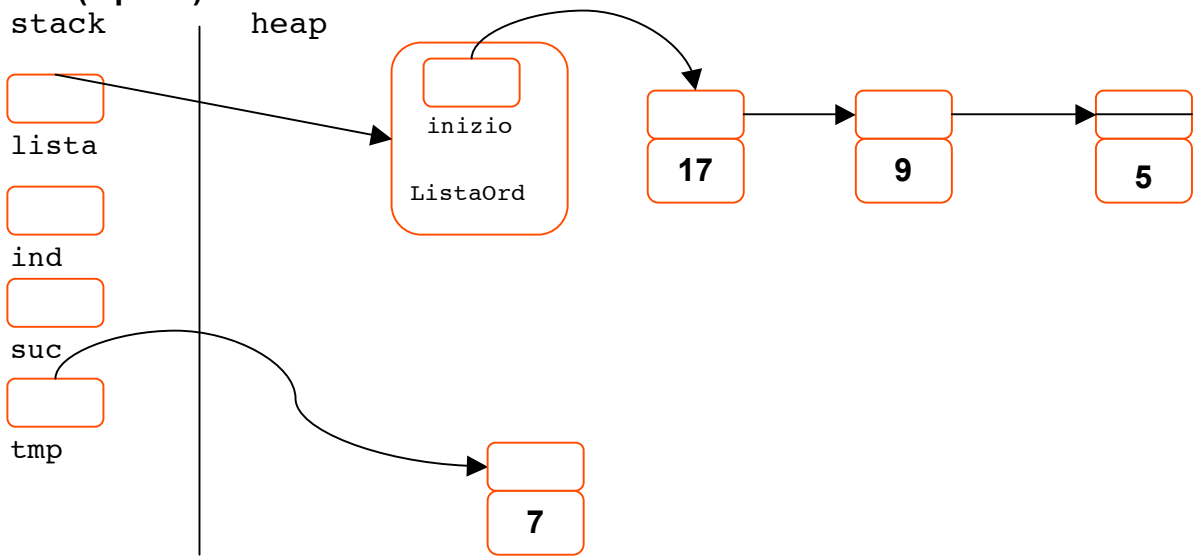
I Appello - 22 gennaio 2009

Cognome **Nome**

Matricola

---

1 (8 punti)



In figura, è schematizzato lo stato della memoria, ad un dato istante, durante il processo di costruzione di una lista ordinata decrescente di numeri, che usa la classe `ListaOrd` con la classe interna `NodoLista`:

```
public class ListaOrd {
    private NodoLista inizio;

    private static class NodoLista {
       Comparable dato;
       NodoLista pros;
    }    ……
}
```

La variabile `lista` sullo stack e' il riferimento alla lista di tipo `ListaOrd` dopo un certo numero d'inserimenti. Si noti che la figura mostra l'usuale semplificazione che mostra nel campo `dato` i valori interi, mentre in realtà dovrebbero esservi i riferimenti ad oggetti: ad esempio di tipo `Integer` o di qualunque classe che implementi l'interfaccia `Comparable`.

1. Definite e inizializzate le due variabili **`ind`** e **`suc`** usate come indice di scorrimento della lista - **`ind`** punta a un generico nodo e `suc` al suo successore.

2. Scrivete le istruzioni necessarie per far avanzare parallelamente gli indici sulla lista:

3. Scrivete la condizione di uscita da un ciclo che serve a far avanzare i due puntatori **ind** e **suc** in modo da prepararsi ad inserire nella lista il nodo **tmp** nella posizione corretta. Si deve riflettere su come far fermare il ciclo in modo da conservare il riferimento al nodo che contiene un dato, inferiore a quello da inserire, e il riferimento al nodo dopo il quale inserire. Le variabili **ind** e **suc** servono allo scopo e consentono di individuare la posizione esatta in cui inserire il nodo **tmp**. Le istruzioni al punto 2. rappresentano il corpo del ciclo.

4. Scrivete le istruzioni necessarie per inserire il nodo **tmp**, creando i legami corretti con i nodi pre-esistenti, una volta che gli indici di scorrimento sulla lista sono giunti alla posizione corretta:

5. Disegnate sulla figura la struttura che si ottiene: i legami che si formano e quelli che si perdono, dopo aver inserito il nodo **tmp** e lo stato delle variabili al termine del ciclo.

---

2 (5 punti)

Disegnate la traccia dell'andamento dello stack delle chiamate dei metodi durante l'esecuzione del metodo **`int f(int, int)`** con valori `f(6, 3)`. Si ricordi che, nel record d'attivazione del metodo i campi `ind` e `val` indicano rispettivamente l'indirizzo e il valore di rientro relativi ad un metodo chiamato. Inoltre si noti che nella figura e' tracciata solo la parte di stack relativa alla prima chiamata del metodo `f` dall'inidirizzo `K` di qualche altro metodo che trascuriamo. La freccia indica la direzione di crescita dello stack

```
int f(int m , int n)     {
  if (m==0&&n==0)
    return 1;
  else if (n==0)
    return 1 + f(m-1, n);      // A    A, B, C sono indirizzi
  else if (m==0)
    return 1 + f(m, n-1);      // B
  else return 2 + f(m-1, n-1) // C
}
```

n 3								
m 6								
val ?								
ind K								

Record 1 Record 2 →

Quale valore riporta la chiamata di `f(6, 3)`?

Quante chiamate ricorsive sono state effettuate, includendo nel conto la I chiamata?

Quale sarebbe invece il valore di `f(3, 6)`?

3 (3 punti)

Definire una struttura dati per archiviare le temperature medie dei diversi mesi degli anni tra il 1980 e il il 1999. La struttura si chiama **`temperature`**; durante la lettura dei dati archiviati si useranno gli indici numerici corrispondenti ai mesi (gennaio 1, febbraio 2 etc).

---

4 (5 punti)

In una classe **`Negozio`** si definisce un campo **`venduti`** che contiene il numero di oggetti venduti in ogni giorno e mese dell'anno. Gli oggetti venduti sono classificati in due categorie, identificate come **`cancelleria`** e **`regali`**.

1. Si definisca la classe **`Oggetti`**, che costituisce il tipo dell'elemento di base della tabella **`venduti`**:

2. Si definisca la tabella **`venduti`** della classe **`Negozio`**

3. Si definisca il metodo **`void init()`** che inizializza opportunamente il campo **`venduti`**, usando le costanti **`GIORNI`** e **`MESE`** dal significato ovvio.

4. Istanziate un oggetto di classe **`Negozio`**, e opportunamente richiamate il metodo **`init()`**

5. Definite ora un ciclo e le eventuali variabili necessarie per calcolare il valore medio degli oggetti di categoria **`cancelleria`** in un dato mese **`mese`**.

---

5 (4 punti)

1. Valutare l'espressione aritmetica: (5 + 3) * (4 / 2) + 4 * 3 =

2. Scrivete l'espressione in modo post-fisso:

3. Valutate l'espressione precedente usando uno stack per la computazione. Disegnate il processo di computazione sullo stack nella griglia proposta, in cui ogni colonna rappresenta una fotografia dello stack a ciascun passo della computazione; il fondo dello stack sia in basso e la cima in alto. La riga in basso nella griglia serve per

evidenziare il token che viene prelevato dall'espressione d'input e in corrispondenza del quale si devono effettuare azioni diverse sullo stack.

6 (4 punti)

Scrivere un ciclo di acquisizione di input per stringhe di caratteri (nomi ad esempio), che dovranno popolare un array. Si supponga che l'input sia su una sola riga in risposta al prompt "**`input?`** ". Si definisca la struttura che immagazzina le stringhe (array) e si usi la classe **`StringTokenizer`**.

7 (3 punti)

Considerate le classi `A` e `B`, estesnsione della classe `A`:

```
class A {
   public int codice;
   protected String tipo;

   A(int c, String t) {
      codice = c;
      tipo = t;
   }
}
```

```
class B extends A {
   private int codice;
   B(int c) {




   }
}
```

1. Definite il costruttore della classe `B` tenedo conto delle seguenti indicazioni:
   a. ogni oggetto di classe `B` è caratterizzato da un numero di codice;
   b. il campo `codice` del suo antenato ha valore pari al valore – 1 del codice dell'oggetto
   c. il campo `tipo` è posto al valore predefinito "`estensione`".
2. Osservate i due membri `codice` definiti nelle due classi. C'è conflitto tra il campo `codice` ereditato e quello definito? SI NO
3. Si dice che il membro `codice` di classe `B` quello di classe `A`
4. Il metodo **`int getCodice() {return super.codice + codice;}`** è corretto per la classe `B`? SI NO
5. Istanziati gli oggetti **`A a = new A(21, "non esteso"); B b = new B(222);`** valutate l'espressione **`b.getCodice()`**