Programmazione e Algoritmi 1

A.A. 2022/23 — Soluzioni appello di prova

prof. Gianluca Amato

Esercizio 1

Vedi file `Esercizio1.java`.

Esercizio 2

Vedi file `Esercizio2.java`.

Esercizio 3

Sia n la lunghezza dell'array in input al meotodo `fillZero`. Il programma consiste di un ciclo `for` che esegue il corpo un numero di volte pari ad $n-1$, ovvero $O(n)$ volte. Il corpo del ciclo `for` non contiene altri cicli, è quindi ha complessià $O(1)$. Ne segue che tutto il programma ha complessità $O(n)$.

Esercizio 4

Vedi file `Esercizio3.java`.

Esercizio 5

Vedi file `Esercizio4.java`.

Esercizio 6

Dopo l'operazione `b += 50`, la variabile `b` dovrebbe assumere il valore 150. Tuttavia, il tipo `byte`, usando una rappresentazione su 8 bit in complemento a 2, può rappresentare solo numeri da -128 a 127: si verifica quindi un *overflow*.

Convertendo 150 in binario, otteniamo 10010110_2, che è già su 8 bit quindi non è necessario aggiungere zeri a sinistra. Tuttavia, sicomme il bit più significativo è 1, in realtà quella sequenza binaria rappresenta un numero negativo. Per determinare di quale numero si tratta, determino il complemento a 2 di 10010110_2, che è 01101010_2, ovvero 106. Dunque, la sequenza di istruzioni indicata stampa il valore -106.

Esercizio 7

Questa è la soluzione dell'esercizio.

riga programma				valore variabili	note
3				args={}	main({})
4				args={}	
	7			a={3,4}	calcola({3,4})
	8			a={3,4}	
	11			a={3,4}	
		16		a={3,4}	dropFirst({3,4})
		17		a={3,4}	
		19		a={3,4}, res={0}	
		20		a={3,4}, res={0}, i=0	
		21		a={3,4}, res={8}, i=0	
		20		a={3,4}, res={8}, i=1	
		23		a={3,4}, res={8}	return {8}
	11			a={3,4}, b={8}	
	12			a={3,4}, b={8}	
		7		a={8}	calcola({8})
		8		a={8}	
		11		a={8}	
			16	a={8}	dropFirst({8})
			17	a={8}	
			19	a={8}, res={}	
			20	a={8}, res={}, i=0	
			23	a={8}, res={}	return {}
		11		a={8}, b={}	
		12		a={8}, b={}	
			7	a={}	calcola({})
			8	a={}	
			9	a={}	return 0
		12		a={8}, b={}	return 8
	12			a={3,4}, b={8}	return 11
4				args={}	print 11

Segue una versione commentata (ma voi per svolgere l'esercizio dovete solo riempiere la tabella, questi commenti sono solo a fini didattici).

Il programma inizia l'esecuzione dal metodo `main`. Se non è specificato diversmente, il parametro di input di `main` (`args`) si assume essere un array di lunghezza zero. Scriviamo nella colonna *riga programma* più a sinistra il valore 3 (la riga di instestazione del metodo `main`), nella colonna *valore variabili* il valore del parametro `args`, e nella colonna *note* la stringa "main({})" per indicare che stiamo invocando il metodo `main` passando come parametro un array vuoto.

riga				valore variabili	note
3				args={}	main({})

L'esecuzione procede alla riga 4. La riga 4 non modifica le variabili, i cui valori sono quindi identici a quelli della riga precedente. Notare che la riga 4 non termina la sua esecuzione immediatamente, perché

chiama il metodo `calcola` con l'array {3,4} come parametro. La indichiamo comunque nel modulo di esecuzione passo passo perché questo ci semplificherà le cose al momento dell'esecuzione del comando `return`.

riga				valore variabili	note
4				args={}	

Notare che nelle note non scriviamo nulla: la colonna *note* viene infatti riempita in una di queste tre occasioni:

1. viene chiamato un metodo: in tal caso il numero di riga è quella contentene l'instestazione del metodo, e la nota contiene il nome del metodo e i parametri attuali;
2. viene eseguita una istruzione return: in tal caso la nota contiene il valore di ritorno;
3. viene eseguita una istruzione di stampa: in tal caso la nota contiene il valore stampato.

Procediamo nell'esecuzione passo-passo come segue:

riga				valore variabili	note
	7			a={3,4}	calcola({3,4})

La riga indicata è la 7 (intestazione del metodo `calcola`), la colonna *valore variabili* contiene il valore iniziale della variabile `a` (che è {3,4} perché questo gli viene passato dal programma principale) e le note contengono, come detto prima, nome del metodo e valori dei parametri attuali.

Notare che il valore 7 è *indentato* di una colonna a destra rispetto alle due linee precedenti. Il livello di indentazione indica il numero di record di attivazione presenti attualmente nello stack (ovvero il numero di chiamate di metodi "in sospeso") ed è utile, quando si esegue una istruzione return, per capire a quale riga tornare nell'esecuzione del programma (troverete maggiori dettagli più avanti).

La prima vera istruzione di `calcola` è alla riga 8. Le variabili non vengono modificate, quindi la colonna *valore variabili* rimane uguale a quella della riga precedente. Successivamente, poiché la condizione dell'istruzione `if` alla riga 8 è falsa, l'esecuzione procede alla riga 11, che invoca a sua volta il metodo `dropFirst`.

riga				valore variabili	note
	8			a={3,4}	
	11			a={3,4}	

A questo punto parte l'esecuzione del metodo `dropFirst`., a cui viene passato come parametro la variabile `a` di `calcola` (che vale {3,4}). Notare l'ulteriore indentazione dei numeri di riga:

riga				valore variabili	note
		16		a={3,4}	dropFirst({3,4})

Adesso viene esguita la riga 17. Essendo la condizione dell'`if` falsa, l'esecuzione va quindi alla riga 19, dove viene creata una variabile *res* di tipo array di interi, che viene inizializzata con un nuovo array di lunghezza `a.length -1`. Siccome `a` è l'array {3,4}, la sua linghezza è 2, quindi `res` avrà lunghezza 1. Tutti i suoi elementi sono inizializzati con il valore 0.

riga				valore variabili	note
		17		a={3,4}	
		19		a={3,4}, res={0}	

Inizia l'esecuzione del ciclo `for`. La prima volta che entriamo nella riga 20 viene creata la variabile `i` ed inizializzata con 0. Siccome la condizione `i < res.length` è vera, l'esecuzione entra nel corpo del ciclo `for`. Alla riga 21, l'istruzione `res[i] = 2 * a[i+1]`, dato il valore di `i`, equivale a `res[0] = 2 * a[1]`, quindi il valore nella posizione 1 dell'array `a` (ovvero 4) viene moltiplicato per 2 e sostituito al valore nella posizione 0 dell'array `res`.

riga				valore variabili	note
		20		a={3,4}, res={0}, i=0	
		21		a={3,4}, res={8}, i=0	

Quindi l'esecuzione torna alla riga 20, dove la variabile `i` viene incrementata. Questa volta, la condizione `i < res.length` è falsa, quindi si esce dal ciclo `for` e si esegue l'istruzione `return`.

riga				valore variabili	note
		20		a={3,4}, res={8}, i=1	
		23		a={3,4}, res={8}	return {8}

Notare, alla riga 23, due cose:

1. prima di tutto, nella colonna *note* annotiamo il fatto che stiamo eseguendo una istruzione `return`, e indichiamo anche il valore di ritorno;

2. inoltre, la variabile `i` non esiste più, perché siamo usciti dal ciclo `for`, e quindi essa non è presente nella colonna *valore variabili*.

Adesso, a causa dell'istruzione `return`, l'esecuzione torna al metodo chiamante, ovvero `calcola`. In particolare, in presenza di ricorsione, potrebbero esserci più istanze dello stesso metodo correntemente sospese. L'indentazione ci aiuta però a trovare quella corretta: è sufficiente individuare, nella nostra traccia di esecuzione, l'ultima riga che ha livello di indentazione inferiore in quello attuale. Nel nostro caso, la riga in questione è questa:

riga				valore variabili	note
	11			a={3,4}	

L'esecuzione quindi riprende da questa riga, con i valori delle variabili indicati in tabella. Nel nostro caso, quello che accade è che viene creata un variabile `b` di tipo array di interi, che viene inizializzata con il risultato del moetodo `dropFirst` appena eseguito, ovvero {8}.

riga				valore variabili	note
	11			a={3,4}, b={8}	

L'esecuzione passo passo procede quindi in questo modo fino a terminazione del programma.