**Esercizi di programmazione per il capitolo 3**

LA SUA PAGINA CONTIENE diversi esercizi per il Capitolo 3 in [Introduzione alla programmazione con Java](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/index.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhidfZzTu7sHSMmrsVmwrJ-4EUsVUw) . Per ogni esercizio viene fornito un collegamento a una possibile soluzione. Ogni soluzione include una discussione su come un programmatore può affrontare il problema e punti interessanti sollevati dal problema o dalla sua soluzione, oltre al codice sorgente completo della soluzione.

**Esercizio 3.1:**

Quante volte devi tirare un paio di dadi prima che escano gli occhi di serpente? Potresti fare l'esperimento arrotolando i dadi a mano. Scrivi un programma per computer che simula l'esperimento. Il programma dovrebbe riportare il numero di tiri che fa prima che i dadi emergano dagli occhi di serpente. (Nota: "Occhi di serpente" significa che entrambi i dadi mostrano un valore di 1.) L' [esercizio 2.2 ha](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c2/ex2-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhi7d2B9yjhc4CzvuS4PoGWqi6ZEpA) spiegato come simulare il lancio di una coppia di dadi.

[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex1-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhi1rmDjJD_c2D6uvbDgZKwjjFABiA)

**Esercizio 3.2:**

Quale numero intero compreso tra 1 e 10000 ha il maggior numero di divisori e quanti divisori ha? Scrivi un programma per trovare le risposte e stampare i risultati. È possibile che diversi numeri interi in questo intervallo abbiano lo stesso numero massimo di divisori. Il tuo programma deve solo stamparne uno. Un esempio nella [sottosezione 3.4.2 ha](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/s4.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhgv2LNYopEKn8HdKOvuPzfRsGd3AQ#control.4.2) discusso i divisori. Il codice sorgente per quell'esempio è [*CountDivisors.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter3/CountDivisors.java) .

Potresti aver bisogno di alcuni suggerimenti su come trovare un valore massimo. L'idea di base è di esaminare tutti gli interi, tenendo traccia del maggior numero di divisori che hai visto *finora* .Inoltre, tieni traccia del numero intero che ha quel numero di divisori.

[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex2-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhg-0IqizbrjD1MLcIFSMUd4WLFzXw)

**Esercizio 3.3:**

Scrivi un programma che valuti espressioni semplici come 17 + 3 e 3.14159 \* 4.7. Le espressioni devono essere digitate dall'utente. L'input è sempre costituito da un numero, seguito da un operatore, seguito da un altro numero. Gli operatori consentiti sono +, -, \* e /. È possibile leggere i numeri con TextIO.getDouble () e l'operatore con TextIO.getChar () . Il tuo programma dovrebbe leggere un'espressione, stamparne il valore, leggere un'altra espressione, stamparne il valore e così via. Il programma dovrebbe terminare quando l'utente inserisce 0 come primo numero sulla linea.

[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex3-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhgddd5AArFxocAvHVmcBY7H2YMKUg)

**Esercizio 3.4:**

Scrivi un programma che legge una riga di testo di input e la suddivide in parole. Le parole dovrebbero essere prodotte una per riga. Una parola è definita come una sequenza di lettere. Tutti i caratteri nell'input che non sono lettere devono essere scartati. Ad esempio, se l'utente inserisce la linea

Ha detto: "Non è una buona idea".

allora l'output del programma dovrebbe essere

lui

disse

Quella

S

non

un

bene

idea

Una versione migliorata del programma elencerebbe "quella" come una singola parola. Un apostrofo può essere considerato come parte di una parola se c'è una lettera su ciascun lato dell'apostrofo.

Per verificare se un personaggio è una lettera, puoi usare (ch> = 'a' && ch <= 'z') || (ch> = 'A' && ch <= 'Z') . Tuttavia, questo funziona solo in inglese e lingue simili. Una scelta migliore è chiamare la funzione standard Character.isLetter (ch) , che restituisce un valore booleano di true se ch è una lettera e false se non lo è. Funziona con qualsiasi carattere Unicode.

[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex4-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhi3bm3E9CuWZsfcMbbBag_GEOr-5Q)

**Esercizio 3.5:**

Supponiamo che un file contenga informazioni sulle cifre di vendita di un'azienda in varie città. Ogni riga del file contiene un nome di città, seguito da due punti (:) seguiti dai dati relativi a quella città. I dati sono un numero di tipo double. Tuttavia, per alcune città, non erano disponibili dati. In queste righe, i dati vengono sostituiti da un commento che spiega perché mancano i dati. Ad esempio, diverse righe del file potrebbero avere il seguente aspetto:

San Francisco: 19887,32

Chicago: nessun rapporto ricevuto

New York: 298734,12

Scrivi un programma che calcolerà e stamperà le vendite totali da tutte le città insieme. Il programma dovrebbe anche riportare il numero di città per le quali i dati non erano disponibili. Il nome del file è "sales.dat".

Per completare questo programma, avrai bisogno di un fatto sull'input di file con *TextIO* che non è stato trattato nella [sottosezione 2.4.4](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c2/s4.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhgyHtc9D7vG2I6Tu0Nvr5GrrTf02g#basics.4.4) . Dato che non sai in anticipo quante righe ci sono nel file, hai bisogno di un modo per dire quando sei arrivato alla fine del file. Quando *TextIO* sta leggendo da un file, la funzione TextIO.eof () può essere utilizzata per verificare la fine del file . Questa funzione con valori booleani restituisce true se il file è stato letto interamente e restituisce false se vi sono più dati da leggere nel file. Ciò significa che puoi leggere le righe del file in un ciclo while (TextIO.eof () == false) .... Il ciclo terminerà quando tutte le righe del file sono state lette.

Suggerimento: per ogni riga, leggi e ignora i caratteri fino al colon. Quindi leggi il resto della riga in una variabile di tipo *String* . Prova a convertire la stringa in un numero e usa try..catch per verificare se la conversione ha esito positivo.

[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex5-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjs2lp-50s23BzACMSolsMIU078Pw)

**Esercizio 3.6:**

[L'Esercizio 3.2](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex2-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhg-0IqizbrjD1MLcIFSMUd4WLFzXw) ti chiedeva di trovare il numero compreso tra 1 e 10000 che abbia il maggior numero di divisori. Dovevi solo stampare uno di questi numeri. Rivedere il programma in modo che stampi **tutti i** numeri con il numero massimo di divisori. Usa un array come segue: Mentre conti i divisori per ciascun numero, memorizza ogni conteggio in un array. Quindi alla fine del programma, puoi passare attraverso la matrice e stampare tutti i numeri che hanno il conteggio massimo. L'output del programma dovrebbe essere simile a questo:

Tra numeri interi compresi tra 1 e 10000,

Il numero massimo di divisori era 64

I numeri con tanti divisori includono:

7560

9240

[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex6-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjIzSNMoDjba21tw9GCkyuIIPnFDg)

**Esercizio 3.7:**

Un esempio nella [sottosezione 3.8.3 ha](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/s8.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhVpjzIR0RD756ECsvf2DPLho-Prg#control.7a.3) cercato di rispondere alla domanda, quante persone a caso devi selezionare prima di trovare una data di nascita doppia? Il codice sorgente per quel programma può essere trovato nel file [*BirthdayProblem.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter3/BirthdayProblem.java) . Ecco alcune domande correlate:

* Quante persone a caso devi selezionare prima di trovare **tre** persone che condividono lo stesso compleanno? (Cioè, tutte e tre le persone sono nate lo stesso giorno nello stesso mese, ma non necessariamente nello stesso anno.)
* Supponi di scegliere 365 persone a caso. Quanti diversi compleanni avranno? (Il numero potrebbe teoricamente essere ovunque da 1 a 365).
* Quante persone diverse devi controllare prima di aver trovato almeno una persona con un compleanno in ciascuno dei 365 giorni dell'anno?

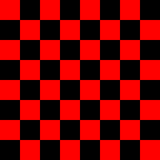
Scrivi **tre** programmi per rispondere a queste domande. Ognuno dei tuoi programmi dovrebbe simulare la scelta casuale delle persone e il controllo dei loro compleanni. (In ogni caso, ignorare la possibilità di anni bisestili).

[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex7-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjpH1DwMJfobQ9xb98r9MBR8uwdAw)

**Esercizio 3.8:**

Scrivi un programma GUI che disegna una scacchiera. Basare la soluzione sul programma di esempio [*SimpleAnimationStarter.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter3/SimpleAnimationStarter.java) , anche se si sta creando solo un'immagine statica piuttosto che un'animazione. Disegnerai la scacchiera nella subroutine drawFrame () . Dovresti leggere i commenti nel file per scoprire altre modifiche che potresti dover apportare.

Supponiamo che la dimensione dell'area di disegno sia 400 x 400 pixel. Una scacchiera contiene 8 righe e 8 colonne di quadrati. Se la dimensione dell'area di disegno è 400, significa che ogni quadrato può essere 50 per 50 pixel. I quadrati sono rossi e neri (o qualsiasi altro colore tu scelga). Ecco un modo complicato per determinare se un dato quadrato deve essere rosso o nero: le righe e le colonne possono essere considerate numerate da 0 a 7. Se il numero di riga del quadrato e il numero di colonna del quadrato sono entrambi pari o entrambi dispari, quindi il quadrato è rosso.Altrimenti, è nero. Si noti che un quadrato è solo un rettangolo in cui l'altezza è uguale alla larghezza, quindi è possibile utilizzare la subroutine g.fillRect () per disegnare i quadrati. Ecco un'immagine ridotta della scacchiera che vuoi disegnare:



[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex8-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhgS4NYob68qvt4XGS3ixyrhtesjNg)

**Esercizio 3.9:**

Spesso, qualche elemento di un'animazione si ripete all'infinito, a tutti i tanti fotogrammi. A volte, la ripetizione è "ciclica", il che significa che alla fine torna indietro. A volte la ripetizione è "oscillante", come un movimento avanti e indietro in cui la seconda metà è uguale alla prima metà riprodotta all'indietro.

Scrivi un'animazione che mostri movimenti ciclici e oscillanti a varie velocità. Per il movimento ciclico, puoi usare un quadrato che si sposta attraverso l'area di disegno, quindi salta indietro all'inizio e poi ripete lo stesso movimento più e più volte. Per il movimento oscillatorio, puoi fare qualcosa di simile, ma il quadrato dovrebbe spostarsi avanti e indietro tra i due bordi dell'area di disegno; cioè, si sposta da sinistra a destra durante la prima metà dell'animazione e quindi retrocede da destra a sinistra durante la seconda metà. Per scrivere il programma, è possibile iniziare con una copia del programma di esempio [*SimpleAnimationStarter.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter3/SimpleAnimationStarter.java) , come nell'esercizio precedente.

Un movimento ciclico deve ripetere ogni fotogramma N per qualche valore di N. Ciò che si disegna in qualche fotogramma dell'animazione dipende dal numero di fotogramma. Il frameNumbercontinua ad aumentare per sempre. Per implementare il movimento ciclico, ciò che si vuole veramente è un "numero di fotogramma ciclico" che assume i valori 0, 1, 2, ..., (N-1), 0, 1, 2, ..., (N- 1), 0, 1, 2, .... È possibile ricavare il valore necessario da frameNumber semplicemente dicendo

cyclicFrameNumber = frameNumber% N;

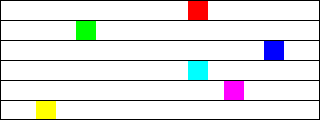
Quindi, basterà basare ciò che si disegna su cyclicFrameNumber anziché su frameNumber . Allo stesso modo, per un'animazione oscillante, è necessario un "numero di frame di oscillazione" che assuma i valori 0, 1, 2, ... (N-1), N, (N-1), (N-2),. .. 2, 1, 0, 1, 2 e così via, ripetendo il movimento avanti e indietro per sempre. Puoi calcolare il valore di cui hai bisogno con

oscilationFrameNumber = frameNumber% (2 \* N);

if (oscillationFrameNumber> N)

oscillationFrameNumber = (2 \* N) - oscillationFrameNumber;

Ecco una schermata della mia versione del programma. Io uso sei quadrati. I primi tre fanno movimento ciclico a varie velocità, mentre i tre inferiori fanno movimento oscillatorio. Ho disegnato linee nere sull'area di disegno per separare i quadrati e dare loro "canali" per entrare.



[Vedi la soluzione](https://translate.googleusercontent.com/translate_c?depth=1&hl=en&rurl=translate.google.it&sl=auto&sp=nmt4&tl=it&u=http://math.hws.edu/eck/cs124/javanotes7/c3/ex9-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhgAKDt3tiQU_N1x4w0AH8uLiGND9Q)