**Esercizi di programmazione per il capitolo 5**

LA SUA PAGINA CONTIENE diversi esercizi per il Capitolo 5 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 5.1:**

In tutte le versioni della classe *PairOfDice* nella [Sezione 5.2](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/c5/s2.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhgUai0QC2YgwAHA3glOkef5F-dluQ) , le variabili di istanza die1 e die2 sono dichiarate pubbliche . Dovrebbero davvero essere privati , in modo da essere protetti dal cambiamento da fuori dalla classe. Scrivi un'altra versione della classe *PairOfDice* in cui le variabili di istanza die1 e die2 sono private . La tua classe avrà bisogno di metodi "getter" che possono essere usati per scoprire i valori di die1 e die2 . (L'idea è di proteggere i loro valori dall'essere modificati al di fuori della classe, ma ancora di consentire la lettura dei valori.) Includere altri miglioramenti nella classe, incluso almeno un metodo toString () . Metti alla prova la tua classe con un breve programma che conta quante volte viene tirata una coppia di dadi, prima che il totale dei due dadi sia uguale a due.

[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/c5/ex1-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhEowVbgQmiY9MgdphJ_BRyCke3hA)

**Esercizio 5.2:**

Un'attività di programmazione comune consiste nel calcolare le statistiche di un insieme di numeri. (Una statistica è un numero che riassume alcune proprietà di un insieme di dati.) Le statistiche comuni includono la media (anche nota come media) e la deviazione standard (che indica come sono distribuiti i dati dalla media). Ho scritto una piccola classe chiamata *StatCalc* che può essere utilizzata per calcolare queste statistiche, nonché la somma degli elementi nel set di dati e il numero di elementi nel set di dati. Puoi leggere il codice sorgente per questa classe nel file [*StatCalc.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/StatCalc.java). Se calc è una variabile di tipo *StatCalc* , sono disponibili i seguenti metodi di istanza:

* calc.enter (item) dove item è un numero, aggiunge l'elemento all'insieme di dati.
* calc.getCount () è una funzione che restituisce il numero di elementi che sono stati aggiunti al set di dati.
* calc.getSum () è una funzione che restituisce la somma di tutti gli elementi che sono stati aggiunti al set di dati.
* calc.getMean () è una funzione che restituisce la media di tutti gli elementi.
* calc.getStandardDeviation () è una funzione che restituisce la deviazione standard degli elementi.

In genere, tutti i dati vengono aggiunti uno dopo l'altro chiamando più e più volte il metodo enter () , man mano che i dati diventano disponibili. Dopo aver inserito tutti i dati, è possibile chiamare uno qualsiasi degli altri metodi per ottenere informazioni statistiche sui dati. I metodi getMean () e getStandardDeviation () devono essere chiamati solo se il numero di elementi è maggiore di zero.

Modificare il codice sorgente corrente, [*StatCalc.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/StatCalc.java) , per aggiungere i metodi di istanza getMax () e getMin () . Il metodo getMax () dovrebbe restituire il più grande di tutti gli elementi che sono stati aggiunti al set di dati e getMin () dovrebbe restituire il più piccolo. Dovrai aggiungere due nuove variabili di istanza per tenere traccia degli oggetti più grandi e più piccoli che sono stati visti finora.

Metti alla prova la tua nuova classe usandola in un programma per calcolare le statistiche per un insieme di numeri diversi da zero immessi dall'utente. Inizia creando un oggetto di tipo *StatCalc* :

Calc StatCalc; // Oggetto da utilizzare per elaborare i dati.

calc = new StatCalc ();

Leggi i numeri dall'utente e aggiungili al set di dati. Utilizzare 0 come valore sentinella (ovvero, interrompere la lettura dei numeri quando l'utente inserisce 0). Dopo aver inserito tutti i numeri diversi da zero dell'utente, stampare ciascuna delle sei statistiche disponibili dal calc .

[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/c5/ex2-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhi3W3W70ydOTauM7aL4jZ6gzyKJLw)

**Esercizio 5.3:**

Questo problema utilizza la classe *PairOfDice* [dell'Esercizio 5.1](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/c5/ex1-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhEowVbgQmiY9MgdphJ_BRyCke3hA) e la classe *StatCalc* [dell'Esercizio 5.2](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/c5/ex2-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhi3W3W70ydOTauM7aL4jZ6gzyKJLw) .

Il programma [dell'Esercizio 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/c4/ex4-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhgx1CVXQsSqXTAruSAo_ii4Vkq3tw) esegue l'esperimento di contare quante volte viene tirata una coppia di dadi prima che un dato totale arrivi. Ripete questo esperimento 10000 volte e poi riporta il numero medio di lanci. Fa tutto questo processo per ogni possibile totale (2, 3, ..., 12).

Ripeti quell'esercizio. Ma invece di segnalare solo il numero medio di lanci, dovresti anche riportare la deviazione standard e il numero massimo di lanci. Usa un oggetto *PairOfDice* per rappresentare i dadi. Utilizzare un oggetto *StatCalc* per calcolare le statistiche. (Avrai bisogno di un nuovo oggetto *StatCalc* per ogni totale possibile, 2, 3, ..., 12. Puoi usare una nuova coppia di dadi se vuoi, ma non è obbligatorio.)

[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/c5/ex3-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhiJ4osdi3kQVZDIQQjf6GDaiB1WiQ)

**Esercizio 5.4:**

La classe *BlackjackHand* della [sottosezione 5.5.1](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/c5/s5.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjKJSgcrCV8FOyAYyU52YGv7ir19w#OOP.5.1) è un'estensione della classe *Hand* della [Sezione 5.4](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/c5/s4.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhioHnjnL3pn5AMZv6S035HVcj_N_w) . I metodi di istanza nella classe *Hand* sono discussi in quella sezione. Oltre a questi metodi, *BlackjackHand* include un metodo di istanza, getBlackjackValue () , che restituisce il valore della mano per il gioco del Blackjack. Per questo esercizio, avrai anche bisogno delle classi *Deck* e *Card* della [Sezione 5.4](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/c5/s4.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhioHnjnL3pn5AMZv6S035HVcj_N_w) .

Una mano di Blackjack contiene in genere da due a sei carte. Scrivi un programma per testare la classe *BlackjackHand* . Dovresti creare un oggetto *BlackjackHand* e un oggetto *Deck* . Scegli un numero casuale tra 2 e 6. Distribuisci molte carte dal mazzo e aggiungile alla mano. Stampa tutte le carte in mano e poi stampa il valore calcolato per la mano da getBlackjackValue () . Ripeti questo finché l'utente vuole continuare.

Oltre a [*TextIO.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/TextIO.java) , il tuo programma dipenderà da [*Card.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/Card.java) , [*Deck.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/Deck.java) , [*Hand.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/Hand.java) e [*BlackjackHand.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/BlackjackHand.java) .

[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/c5/ex4-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhiJGjx0cPdbkF9bKU8Dv_HKFn5Slw)

**Esercizio 5.5:**

Scrivi un programma che consenta all'utente di giocare a Blackjack. Il gioco sarà una versione semplificata del Blackjack in quanto è giocato in un casinò. Il computer fungerà da rivenditore. Come nell'esercizio precedente, il tuo programma avrà bisogno delle classi definite in [*Card.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/Card.java) , [*Deck.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/Deck.java) , [*Hand.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/Hand.java) e [*BlackjackHand.java*](http://math.hws.edu/eck/cs124/javanotes7/source/chapter5/BlackjackHand.java) . (Questo è il programma più lungo e più complesso che sia mai uscito finora negli esercizi).

Dovresti prima scrivere una subroutine in cui l'utente gioca una partita. La subroutine dovrebbe restituire un valore booleano per indicare se l'utente vince il gioco o meno. Restituisce vero se l'utente vince, falso se vince il banco. Il programma richiede un oggetto di classe *Deck* e due oggetti di tipo *BlackjackHand* , uno per il rivenditore e uno per l'utente. L'oggetto generale del Blackjack è ottenere una mano di carte il cui valore è il più vicino possibile a 21, senza andare oltre. Il gioco va così.

* Innanzitutto, vengono distribuite due carte nella mano di ciascun giocatore. Se la mano del banco ha un valore di 21 a questo punto, vince il banco. Altrimenti, se l'utente ha 21, l'utente vince.(Questo è chiamato "Blackjack".) Si noti che il banco vince in pareggio, quindi se entrambi i giocatori hanno Blackjack, vince il banco.
* Ora, se il gioco non è finito, l'utente ha la possibilità di aggiungere alcune carte alla sua mano. In questa fase, l'utente vede le proprie carte e vede **una** delle due carte del mazziere. (In un casinò, il mazziere si offre una carta a faccia in su e una carta a faccia in giù. Tutte le carte dell'utente vengono distribuite a faccia in su.) L'utente prende una decisione se "Hit", che significa aggiungere un'altra carta alla sua mano, o "Stand", che significa smettere di prendere le carte.
* Se l'utente colpisce, c'è la possibilità che l'utente superi 21. In tal caso, il gioco è finito e l'utente perde. In caso contrario, il processo continua. L'utente deve decidere nuovamente se colpire o stare in piedi.
* Se l'utente si trova, il gioco terminerà, ma prima il banco avrà la possibilità di pescare carte. Il rivenditore segue solo le regole, senza alcuna scelta. La regola è che finché il valore della mano del mazziere è inferiore o uguale a 16, il mazziere del mazziere (cioè prende un'altra carta). L'utente dovrebbe vedere tutte le carte del banco a questo punto. Ora, il vincitore può essere determinato: se il banco ha superato 21, l'utente vince. Altrimenti, se il totale del banco è maggiore o uguale al totale dell'utente, vince il banco. Altrimenti, l'utente vince.

Due note sulla programmazione: in qualsiasi punto della subroutine, non appena sai chi è il vincitore, puoi dire " return true ;" o " return false ;" per terminare la subroutine e tornare al programma principale. Per evitare di avere una sovrabbondanza di variabili nella subroutine, ricordare che una chiamata di funzione come userHand.getBlackjackValue () può essere utilizzata ovunque sia possibile utilizzare un numero, incluso in un'istruzione di output o nella condizione di un'istruzione if .

Scrivi un programma principale che consente all'utente di giocare a diversi giochi di Blackjack. Per rendere le cose interessanti, dare all'utente 100 dollari e lasciare che l'utente faccia scommesse sul gioco. Se l'utente perde, sottrarre la scommessa dal denaro dell'utente. Se l'utente vince, aggiungi un importo pari alla scommessa al denaro dell'utente. Terminare il programma quando l'utente vuole smettere o quando finisce i soldi.

[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/c5/ex5-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhh7-h5ExC9xKr4Pcltc-KFeKzfTJQ)

**Esercizio 5.6:**

[L'esercizio 4.7](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/c4/ex7-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhjwx_Vg4OI1wRTyCZ3f8SnucaVzOQ) chiedeva di scrivere un programma che amministra un quiz aggiuntivo di 10 domande. Riscrivi quel programma in modo che usi la seguente classe per rappresentare le domande di aggiunta:

classe pubblica AdditionQuestion {

int privato, b; // I numeri nel problema.

AdditionQuestion public () {// costruttore

a = (int) (Math.random () \* 50 + 1);

b = (int) (Math.random () \* 50);

}

public String getQuestion () {

ritorna "Cosa è" + a + "+" + b + "?";

}

public int getCorrectAnswer () {

restituisce un + b;

}

}

[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/c5/ex6-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhQTmgcViKgxd2GX85JSL3hGIC9FA)

**Esercizio 5.7:**

Riscrivi il programma [dell'esercizio precedente in](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/c5/ex6-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhhQTmgcViKgxd2GX85JSL3hGIC9FA) modo che amministri un quiz con diversi tipi di domande. Nell'esercizio precedente, hai utilizzato una classe per rappresentare le domande di aggiunta. Per questo esercizio, utilizzerai la seguente interfaccia , o una classe astratta equivalente, per rappresentare l'idea più generale di una domanda che ha un numero intero come risposta:

interfaccia pubblica IntQuestion {

public String getQuestion ();

public int getCorrectAnswer ();

}

Puoi fare in modo che la classe *AdditionQuestion* implementa l'interfaccia semplicemente aggiungendo " implements IntQuestion " alla sua definizione. Scrivi una classe simile per rappresentare le domande di sottrazione. Quando si crea un problema di sottrazione, è necessario assicurarsi che la risposta non sia negativa.

Per il nuovo programma, utilizzare un array di tipo IntQuestion [] per contenere le domande del quiz. Includere alcune domande di addizione e alcune domande di sottrazione nel quiz. Puoi anche aggiungere un paio di domande non matematiche, incluso questo, creato come una classe anonima:

IntQuestion bigQuestion = new IntQuestion () {

public String getQuestion () {

torna "Qual è la risposta alla domanda finale" +

"della vita, dell'universo e di tutto?";

}

public int getCorrectAnswer () {

ritorno 42;

}

};

[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/c5/ex7-ans.html&xid=17259,15700002,15700022,15700124,15700149,15700168,15700173,15700186,15700201&usg=ALkJrhg41a2yIZml85kVYi2NG41R54hDOA)