Linguaggio JAVA

Esercizi di programmazione

prof. Roberto FULIGNI
Istituto Tecnico Tecnologico
"Giacomo Fauser" - Novara

Esercizi di programmazione in linguaggio Java

Nelle pagine successive sono indicate le soluzioni degli esercizi che riportano il simbolo ★.

Costrutti di base (sequenza, selezione, iterazione)

1. parallelepipedo ★ Scrivere un programma che, richieste in input lunghezza, larghezza e altezza di un parallelepipedo, ne calcoli la superficie totale e il volume.
2. parcheggio ★ Sapendo che in un parcheggio la prima ora costa 2.50 € mentre tutte le successive costano 1.50 €, scrivere un programma che richieda il numero complessivo delle ore e visualizzi il totale da pagare.
3. biglietto ★ Su una linea ferroviaria, rispetto alla tariffa piena, gli utenti pensionati usufruiscono di uno sconto del 10%, gli studenti del 15% e i disoccupati del 25%. Codificando i pensionati con una P, gli studenti con una S e i disoccupati con una D, scrivere un programma che, richiesto il costo di un biglietto e l'eventuale condizione particolare dell'utente, visualizzi l'importo da pagare.
4. equazione2g ★ Scrivere un programma per risolvere l'equazione di secondo grado $ax^2 + bx + c = 0$, essendo a, b, c coefficienti reali richiesti in input all'utente. Il programma deve comunicare, a seconda dei casi, le due soluzioni oppure la scritta `Non esistono soluzioni reali`.
5. prodcoppie ★ Scrivere un programma che, date n coppie di numeri reali, conti quelle che generano un prodotto negativo, positivo o uguale a zero senza eseguire le moltiplicazioni.
6. mediapd ★ Dati n numeri interi, scrivere un programma che calcoli, quando possibile, la media aritmetica dei valori pari e quella dei valori dispari.

Vettori e matrici

7. mediamaggiori ★ Scrivere un programma che, dopo aver memorizzato in un vettore cinque numeri interi letti da tastiera, visualizzi la media aritmetica *m* e l'elenco degli elementi del vettore maggiori di *m*.
8. vettcasuale ★ Scrivere un programma che memorizzi in un vettore otto numeri interi casuali compresi tra 10 e 100 (estremi inclusi) e li stampi a video sulla stessa riga separandoli con una virgola.
9. indirizzoip ★ Scrivere un programma che generi un indirizzo IPv4 casuale e memorizzi le parti di cui è composto in un vettore di quattro elementi. Successivamente il programma deve visualizzare l'indirizzo IP generato in notazione decimale puntata e riportare la classe di appartenenza.
10. percvoti ★ Scrivere un programma che, dopo aver richiesto l'inserimento da tastiera dei voti di informatica degli alunni di una classe memorizzando i dati in un vettore,visualizzi il voto medio della classe e le percentuali dei voti sufficienti (voto >= 6) e insufficienti.

    Il numero N di alunni deve essere inserito da tastiera. È richiesta la validazione del numero di alunni (1 <= N <= 35) e dei voti immessi (valori interi da 1 a 10). L'output deve essere prodotto secondo il seguente formato:

    `Voto medio: 6,3  Percentuale voti sufficienti: 82,4%  Percentuale voti insufficienti: 13,6%`
11. posmassimo ★ Scrivere un programma che inizializzi un vettore di dimensione N=7 con numeri non necessariamente distinti (lo stesso numero può quindi essere inserito più volte nel vettore), determini il massimo valore memorizzato e lo visualizzi insieme alle posizioni in cui tale valore è presente nel vettore.
12. zigzag – Scrivere un programma che inizializzi staticamente un vettore di *N=12* numeri interi e visualizzi i suoi elementi nel seguente ordine: il primo, l'ultimo, il secondo, il penultimo, il terzo, il terzultimo e così via.

13. **copiainverso** – Scrivere un programma che stampi a video un vettore *va* di *N=15* numeri interi dopo averlo inizializzato con valori casuali da -100 a +100 (estremi inclusi). Successivamente, crea un nuovo vettore *vb* di dimensione pari al numero di valori positivi contenuti in *va* e copia gli elementi positivi di *va* in *vb*. Infine, stampa in contenuto di *vb* in ordine inverso.

14. **consecutivi** ★ Scrivere un programma che inizializzi staticamente e con valori arbitrari un vettore di *N* elementi interi (N > 2) e stampi il messaggio `Tre valori consecutivi coincidenti` se nel vettore sono presenti tre valori uguali in posizioni consecutive, oppure il messaggio `Nessun valore` in caso contrario.

15. **elemdoppio** – Scrivere un programma che, dopo aver inizializzato staticamente un vettore di N elementi interi, mostri il messaggio `Un elemento doppio dell'altro` se nel vettore è presente un elemento doppio di un altro elemento, oppure `Nessun elemento` in caso contrario.

16. **forme** – Una macchina produce N forme di formaggio dal peso nominale di 15 Kg ciascuna. Una forma prodotta è messa in commercio se ha un peso effettivo compreso tra 14.8 e 15.2 kg, altrimenti è scartata.

    Scrivere un programma che richieda l'inserimento dei pesi delle N forme e le memorizzi in un array. Successivamente il programma deve indicare il peso medio delle forme, il numero di forme vendibili e di forme scartate.

17. **stampamatr** ★ Scrivere un programma che legga da tastiera gli elementi interi di una matrice 3x4 (tre righe, quattro colonne e composta da soli numeri positivi minori di 1000) e la visualizzi sullo schermo.

18. **sommarighe** ★ Scrivere un programma che, date due matrici a valori reali inizializzate in fase di dichiarazione e di diversa dimensione, visualizzi le matrici e, per ciascuna di esse, l'indice della riga avente somma massima.

Stringhe

19. **sottostringa** ★ Scrivere un programma che, inseriti da tastiera una stringa non vuota S e due numeri interi non negativi A, B di valore non maggiore della lunghezza di S e A <= B, visualizzi la sottostringa di S compresa tra A e B (estremi inclusi). È richiesta la validazione di S, A e B.

20. **quasiuguali** ★ Scrivere un programma che, lette due stringhe da tastiera, visualizzi il messaggio `UGUALI` se le due stringhe sono uguali, `QUASI UGUALI` se le stringhe differiscono solo per l'uso delle lettere minuscole e maiuscole, `DIVERSE` in caso contrario (le stringhe possono contenere spazi).

21. **mescola** ★ Scrivere un programma che, data una stringa S1, cambi le posizioni dei caratteri di S1 in modo casuale e memorizzi il risultato in una nuova stringa S2. Al termine il programma stampa S1 e S2.

22. **password** ★ I sistemi operativi più recenti richiedono che le password degli utenti soddisfino specifici requisiti di complessità. Consideriamo una password “complessa” se:

    - la sua lunghezza è di almeno sette caratteri;
    - contiene caratteri appartenenti ai seguenti tre gruppi: lettere dell'alfabeto maiuscolo; lettere dell'alfabeto minuscolo; caratteri numerici (0-9)

    Scrivere un programma che generi automaticamente e visualizzi una password complessa casuale di lunghezza non superiore a 12 caratteri.

23. **vocalicons** – Scrivere un programma che prenda una stringa in input e conti il numero di vocali e consonanti presenti nella stringa.

    Esempio:

```
Inserire una stringa: programmazione
Vocali: 6, Consonanti: 8
```

24. **rimuovispazi** – Scrivere un programma che rimuova tutti gli spazi da una stringa data.

    Esempio:

    ```
    Inserire una stringa: Ciao come stai?
    Ciaocomestai?
    ```

25. **contaparole** – Scrivere un programma che conti il numero di parole in una frase, supponendo che le parole siano separate da spazi.

    Esempio:

    ```
    Inserire una stringa: Esercizio di programmazione in Java
    Numero di parole: 5
    ```

26. **parolalunga** – Scrivere un programma che trovi la parola più lunga in una frase, supponendo che le parole siano separate da spazi.

    Esempio:

    ```
    Inserire una stringa: Esercizio di programmazione in Java
    Parola più lunga: programmazione
    ```

27. occorrenze ★ Scrivere un programma che richieda l’inserimento di una stringa da tastiera e visualizzi la tabella delle occorrenze dei suoi caratteri alfabetici.

    Esempio:

    ```
    Inserire una stringa: Ciao, come stai?
    Tabella delle occorrenze
    Lettera     N. occorrenze
       A              2
       C              2
       E              1
       I              2
       M              1
       O              2
       S              1
       T              1
    ```

File di testo

28. txt-numeri ★ Scrivere un programma che memorizzi in un file di testo e su una sola linea i primi 15 numeri pari separati da uno spazio.

29. txt-casuali ★ Scrivere una funzione che, ricevuti in ingresso un nome di file e un numero intero N, memorizzi nel file N numeri casuali disposti a coppie su linee diverse (una coppia per ogni linea).

30. txt-statistica ★ Scrivere un programma che, a partire da uno dei file prodotti nell'esercizio precedente, determini, per ogni coppia, una statistica formata dalla somma e dalla media dei due valori, riportando i risultati nel file

`statistica.txt` (una statistica per ogni linea; all'interno della linea, separare la somma dalla media con un tabulatore).

31. **txt-occorrenze** ★ Scrivere una funzione che, ricevuti in ingresso un nome di file e una parola `S`, restituisca il numero di occorrenze di `S` all'interno del file oppure -1 in caso di errore nella gestione del file. La funzione opera sotto le seguenti condizioni: una parola è una sequenza di caratteri priva di *whitespace*; un file è composto da una sequenza di parole separate da *whitespace*; la ricerca delle occorrenze è di tipo *case insensitive*.

32. **txt-maxmin** ★ Scrivere un programma che calcoli il massimo e il minimo di una sequenza di valori interi contenuti in un file `numeri.txt` e memorizzi i risultati in `maxmin.txt` su due righe diverse nel seguente formato:

    ```
    Valore minimo: ....
    Valore massimo: ....
    ```
    Si ipotizzi di non conoscere a priori né il numero esatto di valori contenuti nel file né il numero massimo di valori previsto per questo esercizio.

33. **cartelle** – Si vuole realizzare un programma che generi automaticamente una cartella per il gioco della tombola. Una cartella è composta da 15 numeri casuali e distinti disposti su tre righe da cinque numeri ciascuna. Il programma richiede innanzitutto il nome del file di testo che conterrà la tabella, quindi genera casualmente i 15 numeri, li memorizza in un'apposita struttura dati e infine memorizza la struttura nel file indicato.

    Esempio di file prodotto:

    ```
    10  87  21  67  81
    76   1  80  34  77
    17  90   8  50  51
    ```

34. **voti1** ★ Il file di testo `voti.txt` contiene i voti di un certo numero di studenti secondo il seguente formato:

    ```
    <cognome>  <nome>  <N>  <voto1>  <voto2> ...  <votoN>
    ```

    Poiché ogni studente può avere un numero diverso di voti, ogni riga include (dopo il nome) il numero N di voti registrati per quello studente, seguito da un elenco di N voti (approssimati a una cifra decimale). Un possibile esempio di file è il seguente:

    ```
    Bianchi Paolo   3   7,0  7,5    6,0
    Rossi   Maria   4   8,5  8,0    8,0    9,0
    Verdi   Carlo   3   5,5  4,5    4,0
    ```
    Scrivere un programma che acquisisca i voti degli studenti e produca in output:

    a) Il file `medie.txt` contenente le medie (approssimate a due cifre decimali) dei voti di ciascuno studente, nel formato:

       ```
       <cognome>  <nome>  <media>
       ```

    b) Il file `esiti.txt` contenente l'esito di fine anno scolastico di ogni studente: "`debito`" se la media è inferiore a 6, "`promosso`" in caso contrario. Il formato del file è:

       ```
       <cognome>  <nome>  <esito>
       ```

35. **sostituzione-file** – Il file di sola lettura `input.txt` contiene un certo numero di linee di testo. Scrivere un programma che, date una stringa non vuota S e una stringa R inserite entrambe da tastiera, sostituisca tutte le occorrenze di S presenti in `input.txt` con la stringa R e memorizzi il risultato finale nel file `output.txt`.

36. **misurecampo** ★ Si vuole realizzare un programma che calcoli il perimetro e l'area di un terreno poligonale. Utilizzando un ricevitore GPS si determinano innanzitutto le coordinate x,y degli N vertici del poligono. I dati sono successivamente memorizzati in un file di testo secondo il seguente formato:

```
u_m
N
x0   y0
x1   y1
...
xN-1   yN-1
```

La prima riga del file contiene una stringa rappresentante l'unità di misura in cui sono espresse le coordinate; la seconda riga contiene un intero `N` che indica il numero dei vertici del poligono; le successive `N` righe contengono le coordinate reali dei vertici (una coppia di coordinate per ogni riga).

Il programma deve svolgere le seguenti operazioni:

a) chiedere all'utente il nome del file di input e acquisirne l'intero contenuto, memorizzando le coordinate in un'apposita struttura dati;

b) determinare il perimetro e l'area del terreno utilizzando le seguenti formule:

perimetro: $P = \sum_{i=0}^{N-1} \sqrt{(x[(i+1) \bmod N] - x[i])^2 + (y[(i+1) \bmod N] - y[i])^2}$ (somma dei lati)

area: $A = \frac{1}{2} \left| \sum_{i=0}^{N-1} (x[i] * y[(i+1) \bmod N] - x[(i+1) \bmod N] * y[i]) \right|$ (formula di Gauss)

c) chiedere all'utente il nome di un file di ouput e memorizzare al suo interno i risultati ottenuti (valore e unità di misura di perimetro e area).

Per esempio, a partire dal seguente file di input:

```
dam
6
-0,57      -1,05
 4,05       8,19
15,60      10,50
24,84       3,57
20,22      -1,05
 8,67      -3,36
```

il programma deve riportare in un nuovo file i messaggi:

```
Perimetro:      61,50 dam
     Area:     229,45 dam^2
```

37. **scrivitab** ★ Scrivere un programma che legga da tastiera e memorizzi in un'apposita struttura dati la tabella:

Cognome	Nome	Altezza (m)	Peso (kg)
Rossi	Mario	1,75	76
Ferraro	Carlo	1,84	82
Marelli	Chiara	1,65	58

Terminato il caricamento, il programma deve memorizzare l'intera tabella nel file di testo `tabella.txt` (un record per linea, campi separati con un tabulatore; non deve essere memorizzata la riga d'intestazione). Si ipotizzi che tutti i campi testuali siano privi di spazi, che le altezze siano numeri *float* e i pesi numeri interi.

38. **mostratab** ★ Scrivere un programma che acquisisca da un file di testo la tabella dell'esercizio precedente e la visualizzi sullo schermo nel seguente formato:

```
+-----------------+-----------------+-------------+-------------+
|     Cognome     |      Nome       | Altezza (m) |  Peso (kg)  |
+-----------------+-----------------+-------------+-------------+
| Rossi           | Mario           |    1,75     |      76     |
| Ferraro         | Carlo           |    1,84     |      82     |
| Marelli         | Chiara          |    1,65     |      58     |
+-----------------+-----------------+-------------+-------------+
```

39. **scrivi-punti** ★ Scrivere un programma che generi casualmente le coordinate x e y di N = 1000 punti del piano cartesiano. Ciascuna coordinata deve essere approssimata a 3 cifre frazionarie e deve appartenere all'intervallo [-10000; +10000). (es. di riga: -32,173; -15,390). Il programma deve richiedere da tastiera un nome di un file di testo e memorizzare in tale file le coordinate generate (una coppia di coordinate separate da un carattere *whitespace* per ciascuna linea).

40. **leggi-punti** ★ Scrivere un programma che richieda l'inserimento di due nomi di file. Successivamente, il programma deve estrarre dal file *nome1* le coordinate memorizzate secondo il formato del problema precedente, determinare la distanza di ciascun punto dall'origine e memorizzare tutte le distanze elaborate (approssimate a 2 cifre decimali) nel file *nome2* (una distanza per ogni linea).

41. **ordina-nomi** – Il file di testo `nomi.txt` contiene l'elenco dei dipendenti di un'azienda, in ordine sparso, nel formato "*Cognome Nome*". Scrivere un programma che memorizzi l'elenco ordinato alfabeticamente nel nuovo file di testo `ordinati.txt` (usare ArrayList e Collections.sort).

Programmazione multithread

42. **caratteri-seq** ★ Scrivere un programma sequenziale che misuri e visualizzi il tempo necessario a visualizzare una qualunque sequenza di 200 caratteri, nell'ipotesi che la stampa di ogni singolo carattere richieda un tempo di 10 ms.

43. **caratteri-par** ★ Risolvere il problema precedente utilizzando un algoritmo parallelo. Utilizzando i tempi rilevati con i due algoritmi, determinare lo *speedup*.

44. **calcoli-matematici** ★ Data la costante numerica N = 1 000 000 000, si considerino le seguenti attività di calcolo:

    a) Determinazione e stampa della somma dei quadrati dei primi N numeri interi positivi: $SQ = 1^2 + 2^2 + 3^2 + \ldots + N^2$

    b) Determinazione e stampa della somma dei logaritmi in base 10 dei primi N numeri interi positivi: $SL = log_{10}(1) + log_{10}(2) + log_{10}(3) + \ldots + log_{10}(N)$

    c) Determinazione e stampa della somma delle radici quadrate dei primi N numeri interi positivi: $SR = sqrt(1) + sqrt(2) + sqrt(3) + \ldots + sqrt(N)$

    Ogni attività termina con la stampa del tempo impiegato per ottenere il risultato (si utilizzino tre "cronometri" distinti).

Realizzare un'applicazione concorrente in Java che esegua le tre attività in parallelo e mostri, alla fine, il tempo totale impiegato.

45. **trevettori** ★ Dati tre vettori di numeri interi VA, VB, VC, ciascuno di lunghezza 20, scrivere un'applicazione che svolga le seguenti attività in modo concorrente:

    a) Attività 1: attende 100 ms, quindi memorizza valori casuali positivi minori di 100 nel vettore VA (seed 2018) e infine stampa su schermo i contenuti dei tre vettori.

    b) Attività 2: attende 200 ms, quindi memorizza valori casuali positivi minori di 50 nel vettore VB (seed 2019) e infine stampa su schermo il più piccolo valore contenuto nei tre vettori.

    c) Attività 3: attende 300 ms, quindi memorizza valori casuai positivi minori di 75 nel vettore VC (seed 2020) e infine stampa su schermo il conteggio di tutti i numeri dispari contenuti nei tre vettori.

46. **codice-seq** – Un programma *open source* scritto in Java consente l'accesso a un'area riservata solo se l'utente digita un codice segreto numerico di sette cifre (la prima cifra non può mai essere zero). Durante l'installazione del software, l'utente sceglie e digita un codice che costituirà la chiave di accesso all'area. Per motivi di sicurezza, il codice-chiave non è memorizzato in chiaro sul disco: il programma trasforma il codice in una stringa complessa denominata *"hash MD5"* e salva sul disco il solo hash.

    Per calcolare l'hash MD5 di una stringa contenente il codice numerico si usa la seguente funzione Java:

```
private static String hashMD5(String strCodice) throws NoSuchAlgorithmException {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update(strCodice.getBytes());
        String hash = String.format("%032x", new BigInteger(1, md.digest()));
        return hash;
}
```

    La funzione rispetta la seguente regola: se due codici sono uguali, allora i loro hash MD5 coincidono; se i codici sono diversi, in generale gli hash non coincidono. In questo modo l'utente può accedere all'area solamente se l'hash del codice inserito è uguale a quello registrato sul disco.

    Purtroppo l'utente ha dimenticato la sequenza segreta e necessita di un programma che ne consenta il recupero. Analizzando il contenuto del disco, si è scoperto che l'hash MD5 del codice-chiave è:

    `ad61e5fa5ca45f1c8f9e40f702160bde`

    Scrivere un programma sequenziale che individui per tentativi (tecnica *brute-force*) il codice segreto confrontando gli hash MD5 di tutti i possibili codici di sette cifre con l'hash noto fino a quando si scopre la sequenza corretta. Determinare inoltre il tempo totale di elaborazione.

47. **codice-par** – Scrivere un programma che risolva il problema precedente in minor tempo utilizzando un opportuno algoritmo parallelo. Determinare il nuovo tempo di elaborazione e confrontarlo con quello dell'esercizio precedente.

Applicazioni proposte

I. All’inizio di un nuovo anno scolastico, un certo istituto pubblica su internet gli elenchi degli studenti suddivisi per classe. La segreteria è in grado di fornire i dati degli alunni memorizzati in un file di testo secondo un formato ben definito. Il file è composto da una parte iniziale in cui sono riportati il nome dell’istituto, l’anno scolastico, il nome del dirigente scolastico e da una tabella contenente la matricola, il cognome, il nome e la classe di tutti gli studenti. La tabella è ordinata per cognome e nome ma non per classe. In questo file, ogni spazio presente nel cognome o nel nome è sostituito con il carattere “_” (underscore).

   Un esempio di file di input è il seguente (tutti i dati sono fittizi, è possibile scaricare dal sito del corso la versione completa del file da 40 classi):

```
   Scuola: Istituto Tecnico "Mario Rossi" - Milano

     A.S.: 2019/2020

Dirigente: dott. Luigi Bianchi


Matricola Cognome Nome Classe

----------------------------

10179 Bento_Da_Silva Greta 3C

10694 Berardi Samuele 3E

10909 Bernacci Alessandro 3A

10036 Bernardi Francesco_Saverio 1G

10050 Bernardi Maria_Rosaria 3D

10150 Bertini Angelica 1E

10594 Bevilacqua Barbara 5B

10811 Bianchi Aldo 5C

continua...
```

   Scrivere un’applicazione che consenta di suddividere per classi il contenuto del file di input memorizzando gli elenchi risultanti in file di testo distinti. Il nome del file di output deve coincidere con quello della classe contenuta (es. 1A.txt).

   All’interno di un file di classe:

   - È presente un’intestazione contenente il nome della scuola, l’anno scolastico e il nome della classe;
   - Gli studenti sono elencati in ordine alfabetico, di ogni studente sono riportati un numero progressivo, il cognome (trasformato in caratteri maiuscoli) e il nome;
   - Ogni carattere “_” è sostituito con uno spazio;
   - Il nome del dirigente scolastico è riportato alla fine dell’elenco.

   La struttura del file di classe è la seguente:

Istituto Tecnico "Mario Rossi" - Milano
Anno scolastico 2019/2020

Classe 3C

1. BENEDETTI Alessandro
2. BENTO DA SILVA Greta
3. CELLINI Enrico
4. CELOTTI Giovanni Pio

...

21. VILLANI Silvia
22. VINCIGUERRA Nadir

Il Dirigente scolastico
(dott. Luigi Bianchi)

Soluzioni degli esercizi

Esercizio n. 1 (parallelepipedo)

```
import java.util.Scanner;

public class Parallelepipedo {

    public static void main(String[] args) {
        double lun, lar, alt;
        double Abase, Alat, Atot;
        double V;

        Scanner s = new Scanner(System.in);
        System.out.println("Inserire le dimensioni del parallelepipedo.");
        System.out.print("Lunghezza: ");
        lun = s.nextDouble();
        System.out.print("Larghezza: ");
        lar = s.nextDouble();
        System.out.print("Altezza: ");
        alt = s.nextDouble();

        Abase = lun * lar;
        Alat = 2 * (lun + lar) * alt;
        Atot = Alat + 2 * Abase;
        V = lun * lar * alt;

        System.out.format("Area totale: %.2f%n", Atot);
        System.out.format("Volume: %.2f%n", V);
    }

}
```

Esercizio n. 2 (parcheggio)

```
import java.util.Scanner;

public class Parcheggio {

    public static void main(String[] args) {
        final double TARIFFA1 = 2.50;       // Tariffe in Euro
        final double TARIFFA2 = 1.50;
        int numOre;
        double tariffa = 0.0;

        Scanner s = new Scanner(System.in);
        do {
            System.out.print("Numero di ore: ");
            numOre = s.nextInt();
        }
        while (numOre < 0);

        if (numOre > 0)
            tariffa = TARIFFA1 + (numOre - 1) * TARIFFA2;

        System.out.format("Tariffa per %d ore: %.2f Euro%n", numOre, tariffa);

    }

}
```

Esercizio n. 3 (biglietto)

```
import java.util.Scanner;
public class Biglietto {
    public static void main(String[] args) {
        final double PERC_SCONTO_PENSIONATI = 0.10;
        final double PERC_SCONTO_STUDENTI = 0.15;
        final double PERC_SCONTO_DISOCCUPATI = 0.25;
        double percSconto;
        double costoIniziale, sconto, costoFinale;
        char categoria;
        Scanner s = new Scanner(System.in);

        System.out.print("Costo iniziale del biglietto (Euro): ");
        costoIniziale = s.nextDouble();

        System.out.println("Indicare la categoria dell'utente.");
        System.out.println("P - Pensionato");
        System.out.println("S - Studente");
        System.out.println("D - Disoccupato");
        System.out.println("Altro tasto - Applicazione della tariffa piena");
        System.out.print("Categoria: ");
        // Legge una stringa da tastiera, la trasforma in maiuscolo
        // ed estrae il primo carattere
        categoria = s.next().toUpperCase().charAt(0);
        switch(categoria) {
            case 'P':
                percSconto = PERC_SCONTO_PENSIONATI;
                break;
            case 'S':
                percSconto = PERC_SCONTO_STUDENTI;
                break;
            case 'D':
                percSconto = PERC_SCONTO_DISOCCUPATI;
                break;
            default:
                percSconto = 0.0;
        }
        sconto = percSconto * costoIniziale;
        costoFinale = costoIniziale - sconto;

        System.out.format("%nCosto iniziale: %.2f Euro%n", costoIniziale);
        System.out.format("Sconto applicato: %.2f Euro%n", sconto);
        System.out.format("Costo finale: %.2f Euro%n", costoFinale);
    }
}
```

Esercizio n. 4 (equazione2g)

```
import java.util.Scanner;

public class Equazione2g {

    public static void main(String[] args) {
        final double TOLLERANZA = 1E-6;
        double a, b, c;
        double delta, x1, x2;
        Scanner s = new Scanner(System.in);

        System.out.println("Inserire i coefficienti dell'equazione a*x^2 + b*x + c = 0");
        do {
            System.out.print("a, b, c:");
            a = s.nextDouble();
            b = s.nextDouble();
            c = s.nextDouble();
        }
        while (a == 0);

        delta = Math.pow(b, 2) - 4.0 * a * c;
        System.out.format("Delta: %g%n", delta);

        if (Math.abs(delta) < TOLLERANZA) {
            // Il delta e' nullo oppure molto piccolo: due soluzioni reali coincidenti
            x1 = -b / (2*a);
            System.out.format("x1 = x2 = %.4f%n", x1);
        }
        else if (delta > 0) {
            // Il delta e' positivo: due soluzioni reali distinte
            x1 = (-b - Math.sqrt(delta)) / (2*a);
            x2 = (-b + Math.sqrt(delta)) / (2*a);
            System.out.format("x1 = %.4f   x2 = %.4f%n", x1, x2);
        }
        else {
            // Delta negativo
            System.out.println("Non esistono soluzioni reali");
        }
    }
}
```

Esercizio n. 5 (prodcoppie)

```
import java.util.Scanner;

public class ProdCoppie {

    public static void main(String[] args) {
        int n;
        int pos = 0, nullo = 0, neg = 0;
        Scanner s = new Scanner(System.in);

        System.out.print("Numero di coppie da inserire? ");
        n = s.nextInt();

        for(int i = 1; i <= n; i++) {
            int a, b;
            System.out.format("Coppia n. %d: ", i);
            a = s.nextInt();
            b = s.nextInt();
            if (a == 0 || b == 0)
                nullo++;
            else if ((a > 0 && b > 0) || (a < 0 && b < 0))
                pos++;
            else
                neg++;
        }
        System.out.format("Numero di coppie a prodotto positivo: %d%n", pos);
        System.out.format("Numero di coppie a prodotto negativo: %d%n", neg);
        System.out.format("Numero di coppie a prodotto nullo: %d%n", nullo);
    }

}
```

Esercizio n. 6 (mediapd)

```
import java.util.Scanner;

public class MediaPD {

    public static void main(String[] args) {
        int n;
        int nPari = 0, sommaPari = 0;
        int nDispari = 0, sommaDispari = 0;
        Scanner s = new Scanner(System.in);

        System.out.print("Numero di valori reali da inserire? ");
        n = s.nextInt();

        for(int i = 1; i <= n; i++) {
            int val;
            System.out.format("Valore n. %d: ", i);
            val = s.nextInt();
            if ((val % 2) == 0) {
                nPari++;
                sommaPari += val;
            }
            else {
                nDispari++;
                sommaDispari += val;
            }
        }

        if (nPari > 0) {
            double m = (double) sommaPari / nPari;
            System.out.format("Media dei valori pari: %.2f%n", m);
        }
        else
            System.out.println("La media dei numeri pari non puo' essere calcolata");

        if (nDispari > 0) {
            double m = (double) sommaDispari / nDispari;
            System.out.format("Media dei valori dispari: %.2f%n", m);
        }
        else
            System.out.println("La media dei numeri dispari non puo' essere calcolata");
    }

}
```

Esercizio n. 7 (mediamaggiori)

```
import java.util.Scanner;

public class MediaMaggiori {

    public static void main(String[] args) {
        final int N = 5;
        int[] v = new int[N];
        int somma = 0;
        double m;
        Scanner s = new Scanner(System.in);

        for(int i = 0; i < N; i++) {
            System.out.format("v[%d]: ", i);
            v[i] = s.nextInt();
            somma += v[i];
        }

        m = (double) somma / N;

        System.out.format("Elementi maggiori della media (%.2f)%n", m);
        for(int i = 0; i < N; i++)
            if (v[i] > m)
                System.out.println(v[i]);
    }
}
```

Esercizio n. 8 (vettcasuale)

```
import java.util.Random;

public class VettCasuale {
    public static void main(String[] args) {
        final int DIM_VETT = 8;
        final int MIN = 10;
        final int MAX = 100;
        int[] v = new int[DIM_VETT];
        Random r = new Random(2019);     // seed 2019

        for(int i = 0; i < DIM_VETT; i++) {
            v[i] = MIN + r.nextInt( (MAX - MIN) + 1);
        }

        System.out.print("v = ");
        for (int i = 0; i < DIM_VETT; i++) {
            if (i > 0)
                System.out.print(", ");

            System.out.print(v[i]);
        }
        System.out.println();
    }
}
```

Esercizio n. 9 (indirizzoip)

```
import java.util.Random;

public class IndirizzoIP {
    public static void main(String[] args) {
        final int DIM_VETT = 4;
        final int MAX = 255;
        int[] ip = new int[DIM_VETT];
        char classe;
        Random r = new Random();

        for(int i = 0; i < DIM_VETT; i++) {
            ip[i] = r.nextInt(MAX + 1);
        }

        for (int i = 0; i < DIM_VETT; i++) {
            if (i > 0)
                System.out.print(".");
            System.out.print(ip[i]);
        }
        System.out.println();

        if (ip[0] < 128)
            classe = 'A';
        else if (ip[0] < 192)
            classe = 'B';
        else if (ip[0] < 224)
            classe = 'C';
        else if (ip[0] < 240)
            classe = 'D';
        else
            classe = 'E';

        System.out.format("Indirizzo di classe %c%n", classe);
    }
}
```

Esercizio n. 10 (percvoti)

```
import java.util.Scanner;

public class PercVoti {
    public static void main(String[] args) {
        final int MAX_N = 35;
        int n;
        int[] voti = new int[MAX_N];
        int somma = 0, contaSuff = 0, contaInsuff = 0;

        Scanner s = new Scanner(System.in);

        do {
            System.out.print("Numero di alunni: ");
            n = s.nextInt();
        }
        while (n < 1 || n > MAX_N);

        for (int i = 0; i < n; i++) {
            int v;
            boolean errore;
            do {
                System.out.format("Voto dell'alunno n. %d: ", i+1);
                v = s.nextInt();

                errore = (v < 1 || v > 10);

                if (errore == true)
                    System.out.print("Voto non valido. ");

            } while (errore == true);
            somma += v;
            if(v >= 6)
                contaSuff++;
            else
                contaInsuff++;
            voti[i] = v;
        }

        System.out.format("Voto medio: %.1f    ", (double) somma / n);
        System.out.format("Percentuale voti sufficienti: %.1f%%    ", 100.0 * contaSuff / n);
        System.out.format("Percentuale voti sufficienti: %.1f%%%n", 100.0 * contaInsuff / n);
    }
}
```

Esercizio n. 11 (posmassimo)

```
public class PosMassimo {
    public static void main(String[] args) {
        int[] vett = {8, 3, 12, 8, 6, 12, 6};
        int max = Integer.MIN_VALUE;

        for (int i = 0; i < vett.length; i++)
            if (vett[i] > max)
                max = vett[i];

        System.out.format("Valore massimo: %d%n", max);
        System.out.println("Posizioni");
        for (int i = 0; i < vett.length; i++)
            if (vett[i] == max)
                System.out.println(i);
    }
}
```

Esercizio n. 14 (consecutivi)

```
import java.util.Scanner;


public class Main {

    public static void main(String[] args) {
            final int[] v = {10, 32, 10, 45, 45, 45, 76, 71};
            int elem = v[0];
            int consecutivi = 1;

        for(int i = 1; consecutivi < 3 && i < v.length; i++) {
            if (v[i] == elem)
                consecutivi++;
            else {
                elem = v[i];
                consecutivi = 1;
            }
        }

        if (consecutivi == 3)
            System.out.println("Tre valori consecutivi coincidenti");
        else
            System.out.println("Nessun valore");
    }
}
```

Esercizio n. 17 (stampamatr)

```
import java.util.Scanner;

public class StampaMatr {
    public static void main(String[] args) {
        final int N = 3;    // Numero di righe
        final int M = 4;    // Numero di colonne
        Scanner s = new Scanner(System.in);

        int[][] m = new int[N][M];

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                int v;
                boolean errore;
                do {
                    System.out.format("m[%d][%d]: ", i, j);
                    v = s.nextInt();

                    errore = (v < 1 || v >= 1000);

                    if (errore == true)
                        System.out.print("Dato non valido. ");

                } while (errore == true);

                m[i][j] = v;
            }
        }

        System.out.println();
        System.out.println("Matrice:");

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++)
                System.out.format("%5d", m[i][j]);

            System.out.println();
        }
    }
}
```

Esercizio n. 18 (sommarighe)

```
public class SommaRighe {

    private static void stampaMatrice(double[][] m) {
        int N = m.length;       // Numero di righe
        int M = m[0].length;    // Numero di colonne

        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++)
                System.out.format("%7.2f", m[i][j]);

            System.out.println();
        }

    }

    private static void stampaIndice(double[][] m) {
        int N = m.length;       // Numero di righe
        int M = m[0].length;    // Numero di colonne
        int i_max = -1;
        double somma_max = Double.NEGATIVE_INFINITY;

        for (int i = 0; i < N; i++) {
            double somma = 0.0;
            for (int j = 0; j < M; j++) {
                somma += m[i][j];
            }
            if (somma > somma_max) {
                i_max = i;
                somma_max = somma;
            }
        }
        System.out.format("%nLa somma e' massima alla riga %d e vale %.2f%n", i_max, somma_max);

    }

    public static void main(String[] args) {

        double[][] m1 = new double[][] {
            {1.4, 4.5, 7.6, 3.2},
            {4.3, 7.5, 3.8, 2.9},
            {5.2, 2.3, 6.0, 4.2}
        };


        double[][] m2 = new double[][] {
            {3.4, 3.2},
```

```
            {5.4, 6.3},
            {10.6, 5.0},
            {9.8, 12.9},
            {8.4, 0.9}
        };

        System.out.println("Matrice m1:");
        stampaMatrice(m1);
        stampaIndice(m1);

        System.out.println();

        System.out.println("Matrice m2:");
        stampaMatrice(m2);
        stampaIndice(m2);
    }

}
```

Esercizio n. 19 (sottostringa)

```
import java.util.Scanner;

public class Sottostringa {

    public static void main(String[] args) {
        String s;
        int a, b;
        Scanner scan = new Scanner(System.in);

        do {
            System.out.print("Inserire una stringa non vuota: ");
            s = scan.nextLine();
        }
        while (s.isEmpty() == true);
        do {
            System.out.format("Inserire a, b (0 <= a <= b <= %d: ", s.length());
            a = scan.nextInt();
            b = scan.nextInt();
        }
        while (a < 0 || a > b || b > s.length());
        String s2 = s.substring(a, b + 1);

        System.out.format("Sottostringa richiesta: %s%n", s2);
    }

}
```

Esercizio n. 20 (quasiuguali)

```
import java.util.Scanner;

public class QuasiUguali {
    public static void main(String[] args) {
        String s1, s2;
        Scanner s = new Scanner(System.in);

        System.out.print("Prima stringa: ");
        s1 = s.nextLine();
        System.out.print("Seconda stringa: ");
        s2 = s.nextLine();

        if (s1.equals(s2))
            System.out.println("UGUALI");
        else if (s1.equalsIgnoreCase(s2))
            System.out.println("QUASI UGUALI");
        else
            System.out.println("DIVERSE");
    }
}
```

Esercizio n. 21 (mescola)

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Mescola {

    private static String mescola(String s) {
        // Genera una lista dei caratteri contenuti nella stringa s.
        List<Character> lista = new ArrayList<>();
        for (char c : s.toCharArray()) {
            lista.add(c);
        }
        StringBuilder sb = new StringBuilder();
        Random r = new Random();

        while (lista.isEmpty() == false) {
            // Sceglie una posizione a caso nella lista
            int i = r.nextInt(lista.size());

            // Rimuove dalla lista il carattere in posizione i
            // e lo copia nella variabile c
            char c = lista.remove(i);

            // Inserisce il carattere rimosso in coda allo string builder
            sb.append(c);
        }

        return sb.toString();
    }

    public static void main(String[] args) {
        String s = "Fauser";

        System.out.format("%s => %s%n", s, mescola(s));
    }

}
```

Esercizio n. 22 (password)

```
import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Password {
    private static Random r = new Random();

    public static void main(String[] args) {
        String pw = generaPassword();
        System.out.println(pw);
    }

    private static String generaPassword() {
        final int LUNG_MIN = 7;
        final int LUNG_MAX = 12;
        final String[] GRUPPI = {
            "ABCDEFGHIJKLMNOPQRSTUVWXYZ",
            "abcdefghijklmnopqrstuvwxyz",
            "0123456789"
        };
        int lunghezza = LUNG_MIN + r.nextInt(LUNG_MAX - LUNG_MIN + 1);
        StringBuilder sb = new StringBuilder();

        // Si garantisce la presenza di almeno un carattere per gruppo
        sb.append(estrai(GRUPPI[0]));
        sb.append(estrai(GRUPPI[1]));
        sb.append(estrai(GRUPPI[2]));
        while (sb.length() < lunghezza) {
            // Sceglie un gruppo a caso
            int ig = r.nextInt(GRUPPI.length);
            sb.append(estrai(GRUPPI[ig]));
        }
        // Mescola i caratteri prima di concludere
        String ris = mescola(sb.toString());

        return ris;
    }

    private static char estrai(String s) {
        // Estrae un carattere a caso dalla stringa s usando
        // il generatore globale r.
        int pos = r.nextInt(s.length());
        return s.charAt(pos);
    }

    private static String mescola(String s) {
```

```
        // Genera una lista dei caratteri contenuti nella stringa s.
        List<Character> lista = new ArrayList<>();
        for (char c : s.toCharArray()) {
            lista.add(c);
        }
        StringBuilder sb = new StringBuilder();
        Random r = new Random();

        while (lista.isEmpty() == false) {
            // Sceglie una posizione a caso nella lista
            int i = r.nextInt(lista.size());

            // Rimuove dalla lista il carattere in posizione i
            // e lo copia nella variabile c
            char c = lista.remove(i);

            // Inserisce il carattere rimosso in coda allo string builder
            sb.append(c);
        }

        return sb.toString();
    }

}
```

Esercizio n. 27 (occorrenze)

```
import java.util.Map;
import java.util.Scanner;
import java.util.TreeMap;

public class Occorrenze {

    public static void main(String[] args) {
        TreeMap<Character, Integer> tabella = new TreeMap<>();
        Scanner scan = new Scanner(System.in);

        String s;

        System.out.print("Inserire una stringa: ");
        s = scan.nextLine().toUpperCase();

        for(char c : s.toCharArray()) {
            if (c >= 'A' && c <= 'Z') {
                if (tabella.containsKey(c) == true) {
                    Integer conta = tabella.get(c);
                    tabella.put(c, conta + 1);
                }
                else
                    tabella.put(c, 1);
            }
        }

        System.out.println("Tabella delle occorrenze");
        System.out.println("Lettera     N. occorrenze");
        for (Map.Entry<Character, Integer> e : tabella.entrySet()) {
            System.out.format("   %c            %4d%n", e.getKey(), e.getValue());
        }
    }

}
```

Esercizio n. 28 (txt-numeri)

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

public class Main {

    public static void main(String[] args) {
        final int N = 15;
        try {
            FileWriter fw = new FileWriter("numeri.txt");
            BufferedWriter bw = new BufferedWriter(fw);
            for(int i = 1; i <= N; i++) {
                bw.write(String.format("%d", 2*i));
                if (i < N)
                    bw.write(' ');
                else
                    bw.newLine();
            }
            bw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Esercizio n. 29 (txt-casuali)

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;

public class Main {
    private static Random r = new Random();
    private static void funzione(String nomeFile, int n) {
        final int MAX_N = 1000;
        if (n % 2 != 0)
            n++;        // Se n è dispari incrementa il suo valore
        try {
            BufferedWriter bw = new BufferedWriter(new FileWriter(nomeFile));
            for(int i = 0; i < (n/2); i++) {
                int n1 = r.nextInt(MAX_N);
                int n2 = r.nextInt(MAX_N);
                bw.write(String.format("%d %d%n", n1, n2));
            }
            bw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
    public static void main(String[] args) {
        funzione("casuali1.txt", 10);
        funzione("casuali2.txt", 20);
    }
}
```

Esercizio n. 30 (txt-statistica)

```
import java.io.*;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
            try (Scanner s = new Scanner(new FileReader("casuali1.txt"));
             BufferedWriter bw = new BufferedWriter(new FileWriter("statistica.txt"))) {
                while (s.hasNext()) {
                    int n1 = s.nextInt();
                    int n2 = s.nextInt();
                    int somma = n1 + n2;
                    double media = somma / 2.0;
                    bw.write(String.format("%d\t%.2f%n", somma, media));
                }
            } catch (FileNotFoundException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
    }
}
```

Esercizio n. 31 (txt-occorrenze)

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

public class Main {

    private static int occorrenze(String nomeFile, String parola) {
        int num = 0;
        try (Scanner s = new Scanner(new FileReader(nomeFile))) {
            while(s.hasNext()) {
                String p = s.next();
                if (p.compareToIgnoreCase(parola) == 0)
                    num++;
            }

        } catch (FileNotFoundException e) {
            e.printStackTrace();
            num = -1;
        }
        return num;
    }
    public static void main(String[] args) {
           Scanner s = new Scanner(System.in);
        System.out.print("Nome del file: ");
        String nome = s.nextLine();
        System.out.print("Parola da cercare: ");
        String parola = s.nextLine();
        int numOcc = occorrenze(nome, parola);
        System.out.format("Numero di occorrenze: %d%n", numOcc);
    }
}
```

Esercizio n. 32 (txt-maxmin)

```
import java.io.*;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int min = Integer.MAX_VALUE;
        int max = Integer.MIN_VALUE;

        try (Scanner s = new Scanner(new FileReader("numeri.txt"))) {
            while (s.hasNext()) {
                int n = s.nextInt();
                if (n < min)
                    min = n;
                if (n > max)
                    max = n;
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        try(BufferedWriter bw = new BufferedWriter(new FileWriter("maxmin.txt"))) {
            bw.write(String.format("Valore minimo: %d%n", min));
            bw.write(String.format("Valore massimo: %d%n", max));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Esercizio n. 34 (voti1)

```
import java.io.*;
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        try (Scanner s = new Scanner(new FileReader("voti.txt"));
            BufferedWriter bwMedie = new BufferedWriter((new FileWriter("medie.txt")));
            BufferedWriter bwEsiti = new BufferedWriter((new FileWriter("esiti.txt")))) {
                while(s.hasNext()) {
                    String cognome = s.next();
                    String nome = s.next();
                    int numVoti = s.nextInt();
                    double somma = 0.0;
                    for(int i = 0; i < numVoti; i++) {
                        double v = s.nextDouble();
                        somma += v;
                    }
                    double media = somma / numVoti;
                    bwMedie.write(String.format("%s %s %.2f%n", cognome, nome, media));
                    bwEsiti.write(
                            String.format("%s %s %s%n",
                                    cognome,
                                    nome,
                                    (media >= 6.0 ? "promosso" : "debito")
                            )
                    );
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Esercizio n. 36 (misurecampo)

```
import java.io.*;
import java.util.Scanner;

public class Main {
     // Campi statici condivisi tra i metodi statici della classe (leggiInput, elaboraDati,
scriviOutput)
    private static String um;
    private static int numVertici;
    private static double[] x;
    private static double[] y;
    private static double perimetro;
    private static double area;

    private static void leggiInput(String nomeFile) {
        try (Scanner s = new Scanner(new FileReader(nomeFile))) {
            // Lettura dei dati e memorizzazione nei campi statici della classe
            um = s.nextLine();
            int n = s.nextInt();
            numVertici = n;
            x = new double[n];
            y = new double[n];
            for(int i = 0; i < n; i++) {
                x[i] = s.nextDouble();
                y[i] = s.nextDouble();
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }

    private static void elaboraDati() {
        int n = numVertici;
        perimetro = 0.0;
        area = 0.0;
        for(int i = 0; i < n; i++) {
            double op1;
            double op2;

            op1 = x[(i+1) % n] - x[i];
            op2 = y[(i+1) % n] - y[i];
            perimetro += Math.sqrt(op1 * op1 + op2 * op2);

            op1 = x[i] * y[(i+1) % n];
            op2 = x[(i+1) % n] * y[i];
            area += op1 - op2;
        }
```

```
        area = 0.5 * Math.abs(area);
    }

    private static void scriviOutput(String nomeFile) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(nomeFile))) {
            // Scrittura dei risultati finali
            bw.write(String.format("Perimetro: %10.2f %s%n", perimetro, um));
            bw.write(String.format("     Area: %10.2f %s^2%n", area, um));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);

        System.out.format("Nome del file di input: ");
        String nomeInput = s.nextLine();
        leggiInput(nomeInput);

        elaboraDati();

        System.out.format("Nome del file di output: ");
        String nomeOutput = s.nextLine();
        scriviOutput(nomeOutput);

    }
}
```

Esercizio n. 37 (scrivitab)

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

class Riga {
    public String cognome;
    public String nome;
    public float altezza;       // [m]
    public int peso;            // [kg]
}

public class Main {

    public static void main(String[] args) {
        int n;
        Riga[] tabella;
        Scanner s = new Scanner(System.in);

        System.out.print("Numero di righe da inserire: ");
        n = Integer.parseInt(s.nextLine());
        tabella = new Riga[n];
        for(int i = 0; i < n; i++) {
            tabella[i] = new Riga();
            System.out.format("Cognome, nome, altezza (m) e peso (kg) della riga n. %d (separati
da spazi): ", i + 1);
            tabella[i].cognome = s.next();
            tabella[i].nome = s.next();
            tabella[i].altezza = s.nextFloat();
            tabella[i].peso = s.nextInt();
        }

        try (BufferedWriter bw = new BufferedWriter(new FileWriter("tabella.txt"))) {
            for(int i = 0; i < n; i++) {
                Riga r = tabella[i];
                     bw.write(String.format("%s\t%s\t%.2f\t%d%n", r.cognome, r.nome, r.altezza,
r.peso));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

Esercizio n. 38 (mostratab)

```
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.util.Scanner;

class Riga {
    public String cognome;
    public String nome;
    public float altezza;       // [m]
    public int peso;            // [kg]
}

public class Main {

    public static void main(String[] args) {

        System.out.println("+-----------------+-----------------+-------------+-------------+");
        System.out.println("|     Cognome     |      Nome       | Altezza (m) |  Peso (kg)  |");
        System.out.println("+-----------------+-----------------+-------------+-------------+");
        try (Scanner s = new Scanner(new FileReader("tabella.txt"))) {
            while(s.hasNext()) {
                Riga r = new Riga();
                r.cognome = s.next();
                r.nome = s.next();
                r.altezza = s.nextFloat();
                r.peso = s.nextInt();
                System.out.format("| %-15s | %-15s |  %6.2f     | %6d      |%n", r.cognome,
                                  r.nome, r.altezza, r.peso);
            }
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
        System.out.println("+-----------------+-----------------+-------------+-------------+");
    }
}
```

Esercizio n. 39 (scrivi-punti)

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;
import java.util.Scanner;

public class Main {
    final static int N = 1000;
    final static double MIN = -10000.0;
    final static double MAX = +10000.0;
    static Random r = new Random(2019);

    private static void generaFile(String nomeFile) {
        try (BufferedWriter bw = new BufferedWriter(new FileWriter(nomeFile))) {
            for (int i = 0; i < N; i++) {
                double x = MIN + (MAX - MIN) * r.nextDouble();
                double y = MIN + (MAX - MIN) * r.nextDouble();
                bw.write(String.format("%.3f\t%.3f%n", x, y));
            }

            System.out.format("File '%s' generato correttamente", nomeFile);

        } catch (IOException e) {
            System.err.format("Errore durante l'elaborazione del file: %s%n", e);
        }
    }

    public static void main(String[] args) {
           String nome;
           Scanner s = new Scanner(System.in);
        System.out.print("Nome del file da generare: ");
        nome = s.nextLine();

        generaFile(nome);
    }
}
```

Esercizio n. 40 (leggi-punti)

```
import java.io.*;
import java.util.Scanner;

public class Main {

    private static void elaboraFile(String nomeInput, String nomeOutput) {
        try (Scanner s = new Scanner(new FileReader(nomeInput));
             BufferedWriter bw = new BufferedWriter(new FileWriter(nomeOutput))
             ) {

            while (s.hasNext() == true) {
                double x = s.nextDouble();
                double y = s.nextDouble();
                double distanza = Math.sqrt(x*x + y*y);

                bw.write(String.format("%.2f%n", distanza));
            }

        } catch (FileNotFoundException e) {
            System.err.format("File non rovato: %s%n", e);
        } catch (IOException e) {
            System.err.format("Errore durante l'elaborazione dei file di output: %s%n", e);
        }
    }

    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        System.out.print("Nome del primo file (file di input): ");
        String nome1 = s.nextLine();
        System.out.print("Nome del secondo file (file di output): ");
        String nome2 = s.nextLine();

        elaboraFile(nome1, nome2);
    }

}
```

Esercizio n. 42 (caratteri-seq)

```
public class Main {

    private static void stampa(char ch) {
        final long ATTESA = 10;    // [ms]
        try {
            Thread.sleep(ATTESA);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.print(ch);
    }
    public static void main(String[] args) {
        final int N = 200;
        long inizio = System.currentTimeMillis();

        // Stampa N/2 caratteri A...
        for(int i = 0; i < N/2; i++) {
            stampa('A');
        }
        // ...E N/2 caratteri B...
        for(int i = 0; i < N/2; i++) {
            stampa('B');
        }
        System.out.println();

        long fine = System.currentTimeMillis();
        long durata = fine - inizio;
        System.out.format("Esecuzione terminata. Tempo impiegato: %d ms%n", durata);
    }
}
```

Esercizio n. 43 (caratteri-par)

```
class CaratteriThread extends Thread {
    int quanti;
    char carattere;

    public CaratteriThread(int n, char ch) {
        quanti = n;
        carattere = ch;
    }

    private void stampa(char ch) {
        final long ATTESA = 10;    // [ms]
        try {
            Thread.sleep(ATTESA);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
        System.out.print(ch);
    }

    @Override
    public void run() {
        for (int i = 0; i < quanti; i++)
            stampa(carattere);
    }
}

public class Main {

    public static void main(String[] args) {
        final int N = 200;

        Thread th1 = new CaratteriThread(N/2, 'A');
        Thread th2 = new CaratteriThread(N/2, 'B');

        long inizio = System.currentTimeMillis();

        th1.start();
        th2.start();

        try {
            th1.join();
            th2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

```
        long fine = System.currentTimeMillis();
        long durata = fine - inizio;

        System.out.println();
        System.out.format("Esecuzione terminata. Tempo impiegato: %d ms%n", durata);

    }
}
```

Esercizio n. 44 (calcoli-matematici)

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class Main {

    public static void main(String[] args) {
        final int N = 100000000;
        System.out.println("TEST SEQUENZIALE");
        System.out.println("------------------------------------------------");
        long ts = testSequenziale(N);

        System.out.println();
        System.out.println("TEST PARALLELO");
        System.out.println("------------------------------------------------");
        long tp = testParallelo(N);
        System.out.format("Speedup: %.2f%n", (double) ts / tp);
    }

    private static void quadrati(int quanti) {
        long inizio = System.currentTimeMillis();
        long sq = 0;
        for(long i = 1; i <= quanti; i++) {
            sq += i* i;
        }
        long fine = System.currentTimeMillis();
        System.out.format("SQ = %d. Tempo di elaborazione: %d ms%n", sq, fine - inizio);
    }

    private static void radici(int quanti) {
        long inizio = System.currentTimeMillis();
        double sr = 0.0;
        for(long i = 1; i <= quanti; i++) {
            sr += Math.sqrt(i);
        }
        long fine = System.currentTimeMillis();
        System.out.format("SR = %f. Tempo di elaborazione: %d ms%n", sr, fine - inizio);
    }

    private static void logaritmi(int quanti) {
        long inizio = System.currentTimeMillis();
        double sl = 0.0;
        for(long i = 1; i <= quanti; i++) {
            sl += Math.log10(i);
        }
        long fine = System.currentTimeMillis();
```

```
        System.out.format("SL = %f. Tempo di elaborazione: %d ms%n", sl, fine - inizio);
    }

    private static long testSequenziale(int n) {
        long inizio = System.currentTimeMillis();
        quadrati(n);
        radici(n);
        logaritmi(n);
        long fine = System.currentTimeMillis();
        long durata = fine - inizio;
        System.out.format("Durata del test sequenziale: %d ms%n", durata);
        return durata;
    }
    private static long testParallelo(int n) {
        long inizio = System.currentTimeMillis();
        ExecutorService executor = Executors.newCachedThreadPool();
        executor.execute(() -> { quadrati(n); });
        executor.execute(() -> { radici(n); });
        executor.execute(() -> { logaritmi(n); });

        executor.shutdown();
        try {
            executor.awaitTermination(Long.MAX_VALUE, TimeUnit.NANOSECONDS);
            long fine = System.currentTimeMillis();
            long durata = fine - inizio;
            System.out.format("Durata del test parallelo: %d ms%n", durata);
            return durata;
        } catch (InterruptedException e) {
            e.printStackTrace();
            return -1;
        }
    }
}
```

Esercizio n. 45 (trevettori)

```
import java.util.Random;
import java.util.concurrent.CountDownLatch;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;

public class Main {
    static int DIM = 20;
    static int[] va = new int[DIM];
    static int[] vb = new int[DIM];
    static int[] vc = new int[DIM];

    static CountDownLatch cdlA = new CountDownLatch(2);
    static CountDownLatch cdlB = new CountDownLatch(2);
    static CountDownLatch cdlC = new CountDownLatch(2);

    static Object lock = new Object();

    private static void stampaVettore(int[] vettore, String nome) {
        System.out.format("%s = [", nome);
        for (int i = 0; i < DIM; i++) {
            if (i > 0) System.out.print(", ");
            System.out.print(vettore[i]);
        }
        System.out.println("]");
    }

    private static void attivita1() {
        Random r = new Random(2018);

        try {
            TimeUnit.MILLISECONDS.sleep(100);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        for (int i = 0; i < DIM; i++) {
            va[i] = 1 + r.nextInt(99);
        }

        cdlB.countDown();
        cdlC.countDown();
        try {
            cdlA.await();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
```

```
        synchronized (lock) {
            stampaVettore(va, "VA");
            stampaVettore(vb, "VB");
            stampaVettore(vc, "VC");
        }
    }

    private static void attivita2() {
        Random r =  new Random(2019);

        try {
            TimeUnit.MILLISECONDS.sleep(200);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        for(int i = 0; i < DIM; i++) {
            vb[i] = 1 + r.nextInt(49);
        }

        cdlA.countDown();
        cdlC.countDown();
        try {
            cdlB.await();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        int m = Integer.MAX_VALUE;

        for (int i = 0; i < DIM; i++) {
            if (va[i] < m) m = va[i];
            if (vb[i] < m) m = vb[i];
            if (vc[i] < m) m = vc[i];
        }

        synchronized (lock) {
            System.out.printf("Valore minimo: %d%n", m);
        }
    }

    private static void attivita3() {
        Random r =  new Random(2020);

        try {
            TimeUnit.MILLISECONDS.sleep(300);
        } catch (InterruptedException e) {
```

```
            e.printStackTrace();
        }

        for(int i = 0; i < DIM; i++) {
            vc[i] = 1 + r.nextInt(74);
        }

        cdlA.countDown();
        cdlB.countDown();
        try {
            cdlC.await();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        int conteggio = 0;

        for (int i = 0; i < DIM; i++) {
            if (va[i] % 2 != 0) conteggio++;
            if (vb[i] % 2 != 0) conteggio++;
            if (vc[i] % 2 != 0) conteggio++;
        }

        synchronized (lock) {
            System.out.printf("Conteggio dei valori dispari: %d%n", conteggio);
        }
    }

    public static void main(String[] args) {
        var e = Executors.newCachedThreadPool();

        e.execute( () -> attivita1() );
        e.execute( () -> attivita2() );
        e.execute( () -> attivita3() );

        e.shutdown();
        try {
            e.awaitTermination(Long.MAX_VALUE, TimeUnit.NANOSECONDS);
        } catch (InterruptedException ex) {
            ex.printStackTrace();
        }

    }
}
```