Capitolo 4

NB: da questo capitolo in poi, ogniqualvolta verrà indicato un valore di input inserito dall'utente si farà riferimento all'utilizzo di un oggetto di tipo *Scanner*. Inoltre, l'utilizzo delle costanti e la scelta del tipo numerico delle variabili, se non diversamente indicato, sarà a discrezione dello studente, non rientrando all'interno dei consigli dell'esercizio.

1. Dato un foglio largo 8.5 pollici e alto 11, convertire e stampare le caratteristiche precedenti in millimetri insieme al suo perimetro e alla diagonale.
   Consiglio: 1 cm corrisponde a 2.54 pollici. Nel calcolo della diagonale tornerà utile utilizzare il metodo *sqrt()* della classe *Math*.
2. Dato un valore fornito in input dall'utente, stampare in output il risultato ottenuto dopo averlo elevato alla seconda, alla terza e alla quarta.
   Consiglio: utilizzare il metodo *pow()* della classe *Math.* Il numero inserito dall'utente potrà contenere una parte decimale.
3. Dati due valori interi forniti in input dall'utente, calcolare e stampare la loro somma, differenza, prodotto, media, distanza, valore massimo e minimo.
   Consiglio: i metodi *abs(), max()* e *min()* della classe *Math* torneranno utili per calcolare rispettivamente la distanza, il valore massimo e quello minimo.
4. Modificare l'esercizio precedente in modo tale che tutte le operazioni il cui risultato è contenuto in un intero siano mostrate in al più 8 byte. L'unica eccezione sarà quindi la media, il cui risultato sarà formattato a video in modo tale da essere mostrato in al più 11 byte, di cui 2 dedicati alla parte decimale.
5. Dato un valore in input fornito dall'utente che rappresenta una distanza in metri, stampare la sua conversione in miglia, piedi e pollici.
6. Dato un valore in input fornito dall'utente indicante il raggio di un cerchio, calcolarne l'area, la circonferenza, il volume e la superficie, stampandole successivamente a video.
   Consiglio: non sarà necessario definire un proprio pi greco. Infatti, la classe *Math* contiene al suo interno la costante statica PI, utile allo scopo dell'esercizio.
7. Dati due valori forniti in input dall'utente che rappresentano la base e l'altezza di un rettangolo, calcolare e stampare a video l'area, il perimetro e la lunghezza della sua diagonale.
8. Un distributore automatico accetta solamente penny e centesimi e fornisce il resto in dollari. Come assunzione, un dollaro corrisponde a cento penny, mentre un centesimo di penny corrisponde a un centesimo di dollaro.
   Creare un distributore automatico che abbia il seguente funzionamento:
   - L'utente dovrà indicare il taglio delle banconote fornite nel resto. Per semplicità, i tagli disponibili sono di 1, 5 e 10 dollari
   - L'utente dovrà indicare il taglio delle monete fornite nel resto. In questo caso,

corrisponderanno sempre ad un quarto di dollaro (1 = $.25, 2 = $.50, 4 = $1)
- L'utente dovrà quindi inserire il prezzo del prodotto scelto, di tipo intero, in penny. Il distributore convertirà tale valore in dollari, stampando a video il resto fornito

9. Creare un programma che stampi a video il costo totale del carburante per un viaggio in auto di 100 miglia e il numero di miglia massime percorribili con il serbatoio attuale. Per ottenere quanto richiesto, l'utente dovrà fornire in input:
    - I litri attuali nel serbatoio
    - L'efficienza dell'auto (km/litro)
    - Il prezzo del carburante per litro
10. Creare una classe *Menu* che rappresenti un menù utente con la seguente struttura:
    - *Attributi:*
        - *labels:* costante stringa statica, conterrà tutte le lettere dell'alfabeto
        - *menuText:* stringa, rappresenterà le voci del menù
        - *optionCount:* intero, rappresenterà il numero di voci nel menù
    - *Costruttore:* dovrà valorizzare gli attributi *menuText* e *optionCount* rispettivamente a stringa vuota e 0
    - *Metodi:*
        - *display():* stampa il contenuto testuale del menù
        - *addOption(String opzione):* aggiunge una voce al menù. La lettera identificativa della nuova voce sarà quella in posizione *optionLabel.* Una volta ottenuta, l'aggiunta della voce al testo corrente, rappresentato dall'attributo *menuText*, dovrà essere fatta secondo il seguente schema:
          {menuText}
          {LetteraDellaNuovaVoce}) {opzione}

          Ai fini di esempio viene fornita un'esecuzione di questa chiamata:

          addOption("pippo"):
          LetteraDellaNuovaVoce = "C"

          menuText =
          "A) Voce Uno
          B) Voce Due
          C) pippo"

    Ai fini di test, creare una classe *MenuDemo* che utilizzi la classe *Menu* aggiungendo varie voci per poi visualizzarle.
11. Scrivere un programma che componga il percorso di un file dati i diversi elementi forniti dall'utente. A tale scopo, all'utente verrà richiesto di:
    - Inserire la lettera del drive in un cui salvare il file
    - Inserire il percorso all'interno del drive che porti alla cartella desiderata
    - Inserire il nome del file

- Inserire l'estensione del file

Come risultato, il programma dovrà stampare il percorso completo del file, ottenuto tramite la concatenazione degli elementi precedenti.

12. Creare un programma che legga una stringa fornita in input dall'utente che rappresenti un numero tra 1000 e 999999 in cui le migliaia sono separate da una virgola (*es:* 1,000 ; 23,456 ; 656,759). Successivamente, stampare il prefisso ed il postfisso di tale numero, definiti rispettivamente come la parte prima e dopo la virgola.
    Consiglio: guardare il metodo *substring()* della classe *String*.
    Esempio: inserendo la stringa 656,759, il programma stamperà in output 656759.
13. Invertire l'esercizio precedente: dato un numero senza virgola fornito in input dall'utente, stampare la versione in cui è stata aggiunta.
    Esempio: inserendo la stringa 656759, il programma stamperà in output 656,759.
14. Stampare a video una griglia come la seguente:

```
+--+--+--+
|  |  |  |
+--+--+--+
|  |  |  |
+--+--+--+
|  |  |  |
+--+--+--+
```

Per ottenere questo risultato, sarà richiesta la creazione di una variabile che rappresenti ciascuna riga della griglia, valorizzata con "+--+--+--+\n| | | |", e di una seconda variabile che rappresenti la riga finale, valorizzata con "+--+--+--+". L'esercizio sarà ritenuto corretto se la stampa di tre righe consecutive seguita da quella della riga finale darà lo stesso risultato dell'esempio.

15. Dato un intero a 5 cifre fornito in input dall'utente, creare una stringa che contenga ogni sua cifra separata da uno spazio.

Esempio: inserendo in input il numero 13567 verrà stampato "1 3 5 6 7".

16. Dati due interi forniti da input che rappresentano due orari nel formato *hhmm,* dove *hh* indica l'ora e *mm* i minuti, stampare la loro differenza in ore e minuti.
    Consiglio: insieme all'esercizio viene fornita la classe *TimeInterval* che servirà a calcolare la differenza tra i due orari.
    NB: il primo orario precede temporalmente il secondo. Pertanto, la differenza tra il valore *0900* e *1730* sarà di 8 ore e 30 minuti, mentre la differenza tra *1730* e *0900* sarà di 15 ore e 30 minuti.
17. Riproporre l'output fornito in *output_17.txt,* che rappresenta la stampa della parola *HELLO* in cui ogni lettera è formata da asterischi e la parola non è in una sola riga. Il corretto svolgimento dell'esercizio vede ogni lettera rappresentata da una corrispettiva variabile.

18. Creare una classe *Balloon* che rappresenti le proprietà di un palloncino.
    - *Attributi:*
        - *volume:* di tipo *double,* rappresenta il volume del palloncino
    - *Costruttore:* dovrà valorizzare l'attributo *volume* a 0
    - *Metodi:*
        - *addAir:* aggiunge dell'aria al palloncino per un valore pari a quello passato alla funzione tramite il suo parametro
        - *getVolume:* ritorna il volume del palloncino
        - *getRadius:* ritorna il raggio del palloncino (sfera), definita come: r = 33volume4
        - *getSurfaceArea:* ritorna l'area della superficie del palloncino, definito come: s = 4r2

    Consiglio: la classe *Math* fornisce un metodo chiamato *cbrt()* per il calcolo della radice cubica e *pow()* per l'elevamento a potenza.

19. Riproporre l'output fornito in *output_19.txt,* che rappresenta la stampa di un albero di Natale. A differenza dell'esercizio precedente, in questo esercizio dovrà essere costruita una classe chiamata *ChristmasTree* con un metodo chiamato *toString()*, il quale restituirà una stringa contenente la rappresentazione grafica dell'albero.
    Per testare il corretto funzionamento, creare una classe chiamata *ChristmasTreePrinter* che istanzierà un oggetto di tipo *ChristmasTree* e stampando a video il risultato ottenuto dal metodo *toString()*.

20. Creare una classe chiamata *IceCreamCone* che simula la creazione di un cono gelato.
    - *Attributi:*
        - *height:* di tipo double, indica l'altezza del cono
        - *radius:* di tipo double, indica il raggio della parte circolare del cono
    - *Costruttore:* dovrà essere ridefinito per accettare due parametri che andranno a valorizzare *height* e *radius*
    - *Metodi:*
        - *getSurfaceArea():* ritorna la superficie laterale del cono, definita come: *Sl = r a, dove a = r2+ h 2*
        - *getVolume():* ritorna il volume del cono, definita come: *V = (r2h)3*

    Per testare il corretto funzionamento, creare una classe chiamata *IceCreamTester* il quale dovrà creare un cono gelato di altezza *6* e raggio *1*. La superficie risultante dovrà essere pari a *19.109562*, mentre il volume dovrà essere pari a *6.283185.*