Il file contiene i testi dei progetti Java tra cui scegliere quello da portare all'esame. Il progetto scelto **non** deve essere il medesimo portato ad un esame di un altro diverso corso con lo stesso docente.

Lo studente è tenuto a far pervenire al docente via e-mail o di persona, su idoneo supporto informatico, il progetto Java **completo** almeno **quattro giorni prima** della data dell'esame, indicando espressamente l'appello a cui intende presentarsi per sostenere l'esame stesso.

I progetti sono di varia difficoltà e lo studente è invitato a scegliere tra essi in base alle proprie capacità ed ambizioni.

E'consentito modificare i progetti, aggiungendo ad esempio funzionalità migliorative o proponendo varianti (**sensate**) che non ne alterino la sostanza. La richiesta di gestire file di testo va in genere intesa come opzionale, seppure migliorativa, del progetto e può essere generalmente sostituita da una gestione con strutture diverse, quali ad esempio array di oggetti.

6.31 I computer giocano un ruolo sempre più importante nel campo educativo. Scrivete un programma che aiuti uno studente di scuola elementare ad apprendere la moltiplicazione. Utilizzate **`Math.random`** per produrre due interi positivi composti da una cifra. Dovrete quindi visualizzare una domanda come:

```
Quanto fa 6 per 7?
```

In seguito lo studente digiterà la risposta e il programma la controllerà. Se è corretta, visualizzate `"Molto bene!"` e sottoponete un'altra domanda; se è sbagliata, visualizzate `"No. Riprova."` e lasciate che lo studente provi ancora la stessa domanda, finché non avrà risposto correttamente. Per generare ogni nuova domanda, deve essere usato un metodo separato; questo metodo deve essere chiamato una volta quando l'applet inizia l'esecuzione, e poi ogni volta che l'utente risponde correttamente alla domanda. Tutti i disegni dell'applet devono essere eseguiti dal metodo **`paint`**.

6.32 L'utilizzo dei computer nel campo educativo è detto *computer-assisted instruction* (CAI). Uno dei problemi che si presenta negli ambienti CAI è l'affaticamento dello studente. Questo problema può essere evitato variando il dialogo del computer, in modo da mantenere viva l'attenzione dello studente. Modificate il programma dell'esercizio 6.31 in modo che siano visualizzati vari commenti a fronte di ogni risposta corretta e sbagliata, come segue:

Commenti per una risposta corretta

```
Molto bene!
Ottimo!
Complimenti!
Continua cosi'!
```

Commenti per una risposta sbagliata

```
No. Riprova.
Sbagliato. Prova ancora.
Non rinunciare!
No. Continua a provare.
```

Utilizzate la generazione di numeri casuali per sceglierne uno compreso tra 1 a 4 e selezionare un commento appropriato a ogni risposta. Utilizzate una struttura **`switch`** nel metodo **`paint`** per emettere i commenti.

6.33 I sistemi più raffinati di istruzione assistita dal computer monitorano le prestazioni dell'utente durante un certo periodo di tempo. La decisione di passare a un nuovo argomento è spesso basata sul successo dello studente con gli argomenti precedenti. Modificate il programma dell'esercizio 6.32 in modo da contare il numero di risposte corrette e sbagliate immesse dallo studente. Dopo che lo studente ha digitato 10 risposte, il programma deve calcolare la percentuale di quelle corrette. Se la percentuale è inferiore al 75 percento, il programma deve visualizzare "**`Chiedi aiuto all'insegnante`**" e quindi terminare la propria esecuzione.

7.27 (*Il crivello di Eratostene*) Un numero primo è un intero divisibile soltanto per 1 e per sé stesso. Il Crivello di Eratostene è un metodo per trovare i numeri primi. Funziona così:

a) Create un array con tutti gli elementi inizializzati a 1 (`true`). Gli elementi dell'array che hanno per indice dei numeri primi resteranno uguali a 1. Tutti gli altri, invece, verranno impostati a zero.
b) Partite dall'indice 2 (l'indice 1 è necessariamente primo) e, ogni volta che trovate un elemento dell'array uguale a 1, impostate a zero tutti gli elementi il cui indice è multiplo dell'elemento trovato. Per l'elemento di indice 2, dovremo azzerare tutti gli elementi che si trovano dopo 2 e che sono multipli di due, cioè tutti gli elementi di indice pari; ugualmente, per l'elemento di indice 3, dovremo azzerare tutti gli elementi che hanno per indice un suo multiplo, come 6, 9, 12, 15, e così via.

Quando avremo completato il procedimento, gli elementi dell'array che valgono ancora 1 avranno per indice un numero primo. Scrivete un programma che implementi il crivello di Eratostene su 1000 elementi, per determinare i numeri primi compresi tra 1 e 999. Ignorate l'elemento 0 dell'array.

7.28 (*Un'altra tecnica di ordinamento: bucket sort*) Questa tecnica di ordinamento prevede l'utilizzo di un array unidimensionale di interi positivi da ordinare, e un array bidimensionale di interi. In questo array le righe hanno indici che vanno da 0 a 9 e le colonne hanno indici che vanno da 0 a *n*-1, dove *n* è il numero di valori da ordinare. Ogni riga dell'array bidimensionale è detta *bucket* o *secchiello*. Scrivete un metodo **`bucketSort`**, che prende come argomento un array di interi ed esegue le seguenti operazioni:

a) Pone ogni valore dell'array unidimensionale in una riga dell'array di bucket, sulla base della cifra delle unità presente nel numero. Per esempio, il valore 97 viene posto nella riga 7, il valore 3 nella riga 3 e il valore 100 nella riga 0. Questo procedimento è chiamato "passaggio di distribuzione".

b) Per mezzo di un ciclo tra le righe dell'array bucket copiate nuovamente i valori nell'array originario. Questo procedimento è chiamato "passagio di raccolta". Il nuovo ordine dei valori nell'array unidimensionale è 100, 3 e 97.

c) Ripetete il procedimento per ogni successiva cifra (decine, centinaia, migliaia ecc.).

Al secondo passaggio, 100 viene posto nella riga 0, 3 viene posto nella riga 0 (perché 3 non ha la cifra delle decine) e 97 nella riga 9. Dopo il passaggio di raccolta, l'ordine dei valori nell'array unidimensionale è 100, 3 e 97. Al terzo passaggio, 100 è posto nella riga 1, 3 nella riga 0 e 97 nella riga 0 (dopo il 3). Dopo l'ultimo passaggio di raccolta, gli elementi dell'array originale sono in ordine.

Notate che l'array bidimensionale dei bucket è grande dieci volte l'array da ordinare. Questa tecnica di ordinamento è più efficiente rispetto all'ordinamento a bolle, ma richiede molta più memoria. Nell'ordinamento a bolle, lo spazio richiesto è di un solo dato aggiuntivo. Questo tipo di ordinamento copia tutti i dati nuovamente nell'array originario a ogni passaggio. Un'altra possibilità è quella di creare un secondo array bidimensionale di bucket e scambiare ripetutamente i dati tra i due array.

8.2 Create una classe **Complex** per effettuare operazioni matematiche su numeri complessi. Scrivete un programma driver per testare la vostra classe. I numeri complessi hanno questa forma

```
realPart + imaginaryPart * i
```

dove i è $\sqrt{-1}$

Rappresentate i dati **private** della classe tramite variabili a virgola mobile. Fornite un costruttore che inizializzi ogni oggetto della classe quando è dichiarato. Fornite un costruttore senza argomenti che contenga dei valori di default nel caso in cui siano forniti degli inizializzatori. Fornite dei metodi **public** per ognuna delle seguenti operazioni:

a) Addizione di due numeri **Complex**: parti reali e parti immaginarie vanno sommate separatamente.
b) Sottrazione di due numeri **Complex**: la parte reale dell'operando destro è sottratta dalla parte reale dell'operando sinistra, e lo stesso vale per le parti immaginarie.
c) Visualizzazione di numeri **Complex** nella forma **(a, b)** dove **a** è la parte reale e **b** quella immaginaria.

8.3 Create una class **Rational** per effettuare operazioni matematiche sulle frazioni. Scrivete un programma driver per testare la vostra classe.

Rappresentate i dati **private** della classe con variabili interé (**numerator** e **denominator**). Il costruttore deve inizializzare ogni oggetto della classe quando viene dichiarato, e deve contenere dei valori di default, nel caso non siano forniti inizializzatori. Il costruttore deve memorizzare i valori delle frazioni in forma ridotta; per esempio, la frazione 2/4 sarà memorizzata come 1 nel **numerator** e 2 nel **denominator**. Fornite dei metodi **public** per ognuna delle seguenti operazioni:

a) Addizione di due numeri **Rational**. Il risultato è memorizzato in forma ridotta.
b) Sottrazione di due numeri **Rational**. Il risultato è memorizzato in forma ridotta.
c) Moltiplicazione di due numeri **Rational**. Il risultato è memorizzato in forma ridotta.
d) Divisione di due numeri **Rational**. Il risultato è memorizzato in forma ridotta.

e) Visualizzazione dei numeri **Rational** nella forma **a/b**, dove **a** è il **numerator** e **b** è il **denominator**.
f) Visualizzazione dei numeri **Rational** nel formato a virgola mobile. (Fornite delle capacità di formattazione che permettano all'utente della classe di specificare il numero di cifre a destra della virgola decimale.)

7.41 (*Simulazione: la lepre e la tartaruga*) In questo problema dovete ricreare la classica gara fra la lepre e la tartaruga. Utilizzerete i numeri casuali per la simulazione di questo evento.

I concorrenti iniziano la gara dal "riquadro 1", il primo di 70 riquadri. Ogni riquadro è una posizione possibile lungo il percorso della gara. La bandierina finale si trova sul riquadro 70. Il primo concorrente che raggiunge o sorpassa il riquadro 70 vince un succulento cestino pieno di carote e lattuga.

Il percorso si snoda lungo il versante in salita di una collinetta, per cui capita che uno dei due concorrenti possa perdere un po' di terreno. Un cronometro incrementa il suo valore di 1 tic al secondo. Ad ogni tic del cronometro il programma aggiorna la posizione degli animali, in base alle seguenti regole.

Animale	Mossa	% di tempo	Spostamento
Tartaruga	Passo veloce	50%	3 riquadri a destra
	Scivolata	20%	6 riquadri a sinistra
	Passo lento	30%	1 riquadro a destra
Lepre	Riposo	20%	Nessuno spostamento
	Salto lungo	20%	9 riquadri a destra
	Scivolata lunga	10%	12 riquadri a sinistra
	Salto piccolo	30%	1 riquadro a destra
	Scivolata piccola	20%	2 riquadri a sinistra

Utilizzate delle variabili per tenere traccia delle posizioni degli animali (ovvero del quadrato su cui si trovano, nell'intervallo 1-70). Fate partire ogni animale dalla posizione 1. Se un animale scivola ancora prima del riquadro 1, abbiate pietà e rimettetelo nuovamente sul riquadro 1.

Generate le percentuali della tabella precedente con un numero intero casuale i nell'intervallo $1 \pounds i \pounds 10$. Per la tartaruga, fatela andare a passo veloce quando $1 \pounds i \pounds 5$, fatela scivolare quando $6 \pounds i \pounds 7$, fatela andare a passo lento quando $8 \pounds i \pounds 10$. Usate una tecnica simile per spostare la lepre.

Ogni gara che si rispetti comincia con lo sparo iniziale. Visualizzate quindi

```
BANG !!!!!

PARTITI !!!!!
```

Per ogni tic del cronometro (ovvero a ogni iterazione del ciclo), visualizzate una linea con 70 tacche, dove mostrate la posizione della tartaruga con una `T` e quella della lepre con una `L`. Può anche capitare che i due concorrenti capitino sullo stesso riquadro. In questo caso, la tartaruga morde la lepre e il programma visualizzerà un `OUCH!!!` in quella posizione. Tutti gli spazi dove non c'è né la `T`, né la `L` e neppure `OUCH!!!` dovrebbero restare vuoti.

Dopo avere visualizzato ogni linea, controllate se uno degli animali ha raggiunto o sorpassato il riquadro 70. Se è così, visualizzate il nome del vincitore e terminate la simulazione. Se vince la tartaruga visualizzate `HA VINTO LA TARTARUGA!!! YAHOO!!!` Per par condicio, naturalmente, utilizzate una scritta simile anche se vince la lepre. E se entrambi raggiungono il riquadro 70 nello stesso momento? Fate come credete. Se volete potete favorire in ogni caso la tartaruga, oppure chiedere una rivincita. Se nessuno degli animali vince, effettuate il ciclo di nuovo per il successivo tic del cronometro.

8.17 Create la classe **IntegerSet**. Ogni oggetto di questa classe può contenere degli interi compresi nell'intervallo da 0 a 100. Un insieme di interi (*set*) è rappresentato internamente da un array di **boolean**. L'elemento dell'array **a[i]** è **true** se l'intero *i* si trova nell'insieme di interi; l'elemento dell'array **a[j]** è **false** se l'intero *j* non si trova nell'insieme di interi. Il costruttore senza argomenti inizializza un insieme "vuoto", ovvero crea un insieme la cui rappresentazione dell'array contiene tutti valori **false**. Fornite i seguenti metodi: il metodo **unionOfIntegerSets** crea un terzo insieme che è

l'unione dei due insiemi esistenti (un elemento dell'array del terzo insieme è impostato a **true** se questo elemento è **true** in uno o entrambi gli insiemi esistenti, altrimenti è impostato a **false**). Il metodo **intersectionOfIntegerSets** crea un terzo insieme che rappresenta l'intersezione dei due insiemi esistenti (un elemento dell'array del terzo insieme è impostato a **false** se questo elemento è **false** in uno o entrambi gli insiemi esistenti, altrimenti è **true**). Il metodo **insertElement** inserisce un nuovo intero *k* in un insieme (impostando **a[k]** a **true**). Il metodo **deleteElement** cancella l'intero *m* (impostando **a[m]** a **false**). Il metodo **setPrint** visualizza un elenco di numeri separati da spazi. Visualizzate soltanto gli elementi che sono presenti nell'insieme. Visualizzate **---** per un insieme vuoto. Il metodo **isEqualTo** determina se due insiemi sono uguali. Scrivete un programma per testare la classe **IntegerSet**. Istanziate vari oggetti **IntegerSet**. Verificate che tutti i vostri metodi funzionino correttamente.

10.26 (*Codice Morse*) Il codice Morse assegna una serie di punti e trattini a ogni lettera dell'alfabeto, a ogni numero e ad alcuni caratteri speciali (come il punto, la virgola, i due punti e il punto e virgola). Nei sistemi basati sui suoni, il punto rappresenta un suono breve, mentre il trattino un suono più prolungato. La separazione delle parole è indicata da uno spazio o, più semplicemente, dall'assenza di un punto o di un trattino. Nei sistemi basati sui suoni, lo spazio è indicato da un breve periodo di tempo, durante il quale non viene trasmesso alcun suono. La figura 10.22 mostra la versione internazionale del Codice Morse.

Scrivete un'applicazione che legga una frase e la traduca in codice Morse. Scrivete inoltre un programma che legga una frase in codice Morse e la traduca in italiano. Usate uno spazio bianco tra ogni lettera del codice Morse e tre spazi bianchi tra ogni parola.

Carattere	Codice	Carattere	Codice	Carattere	Codice
A	.-	J	.---	S	...
B	-...	K	-.-	T	-
C	-.-.	L	.-..	U	..-
D	-..	M	--	V	...-
E	.	N	-.	W	.--
F	..-.	O	---	X	-..-
G	--.	P	.--.	Y	-.--
H	....	Q	--.-	Z	--..
I	..	R	.-.		
Numeri					
1	.----				
2	..---				
3	...--				
4	....-				
5	.....				
6	-....				
7	--...				
8	---..				
9	----.				
0	-----				

Figura 10.22 Il codice Morse

10.24 (*Protezione degli assegni*) I computer vengono frequentemente utilizzati nei sistemi di emissione di assegni, soprattutto da parte delle aziende. Per evitare che gli importi da stampare sugli assegni vengano alterati, però, molti di questi sistemi utilizzano una tecnica di protezione. Gli assegni progettati per essere compilati da un computer contengono un numero fisso di spazi in cui il computer può stampare un importo. Supponendo che un assegno destinato a pagare uno stipendio in dollari contenga otto spazi bianchi in cui il computer deve stampare l'importo di uno stipendio settimanale, se l'importo è di grandi dimensioni, verranno riempiti tutti e otto gli spazi, come nel seguente esempio:

```
1,230.60 (importo dell'assegno)
--------
12345678 (numeri delle posizioni)
```

Se, invece, l'importo è inferiore a 1000 dollari, alcuni degli spazi vengono normalmente lasciati vuoti, come nel seguente esempio, che contiene tre spazi vuoti:

```
  99.87 (importo dell'assegno)
--------
12345678 (numeri delle posizioni)
```

Se viene stampato un assegno con degli spazi vuoti, è facile che qualcuno possa alterarne l'importo. Per impedire che un assegno possa essere alterato, molti sistemi inseriscono degli asterischi al posto degli spazi vuoti, come nel seguente esempio:

```
***99.87
--------
12345678
```

Scrivete un'applicazione che immetta un importo in dollari da stampare su di un assegno e che poi mostri l'assegno in formato protetto, con gli asterischi se necessario. Fate in modo di avere nove spazi disponibili per stampare l'importo.

10.25 (*Scrivere in parole l'importo di un assegno*) Per motivi di sicurezza, l'importo di un assegno deve normalmente essere scritto sia in numeri che in lettere. Molti sistemi computerizzati per l'emissione degli assegni non stampano l'importo anche in lettere, forse perché la maggior parte dei linguaggi usati nelle applicazioni di tipo commerciale non offrono sufficienti funzionalità per la manipolazione delle stringhe. Scrivete un'applicazione che immetta un importo numerico e ne scriva l'equivalente in lettere.

10.27 (*Un programma di conversione delle unità di misura*) Scrivete un'applicazione che assista l'utente durante le conversioni delle unità di misura. Il programma deve permettere all'utente di specificare i nomi delle unità sotto forma di stringhe (centimetri, metri, litri, grammi, pollici, quarti, once, ecc.) e deve rispondere a domande di questo tipo:

```
"Quanti pollici ci sono in 2 metri?"
"Quanti litri ci sono in 10 quarti?"
```

Il programma dovrebbe inoltre riconoscere le conversioni non valide. Per esempio, la domanda:

```
"Quanti piedi ci sono in 5 chilogrammi?"
```

non ha senso, perché i piedi sono un'unità di lunghezza mentre i chilogrammi sono un'unità di peso.

4 Scrivere una classe che confronti due file e determini se sono uguali o meno.

5 Scrivere una classe che permetta di concatenare due o più file.

6 Scrivere una classe che permetta di copiare tutti i file di una directory in un'altra.

7 Scrivere una classe che permetta di modificare il nome di un file.

8 Scrivere una classe che permetta di sostituire una parola all'interno di un file con un'altra di uguale lunghezza.

NB: due a scelta con interfaccia utente.

Si supponga di avere un file di testo contenente un certo numero di parole di senso compiuto.
Si scriva un programma che scelga a caso una delle parole contenute nel file di testo e proponga all'utente di indovinare tale parola.
L'utente tenterà quindi di indovinare, una alla volta, le lettere contenute nella parola. Ogni volta che viene indovinata una lettera, questa verrà visualizzata nella posizione corretta all'interno della parola stessa. Se invece la lettera proposta dall'utente non appartiene alla parola, si avrà un errore. Sarà consentito all'utente di compiere soltanto un numero limitato di errori.

Dato un mazzo di carte da gioco, si calcoli (manualmente) quale è la possibilità che in un'estrazione a caso vengano scelte due carte dello stesso seme. Si implementi un programma che realizzi una simulazione ripetuta di tale estrazione e si verifichi se dopo un certo numero di tentativi la frequenza di tale evento è vicina alla probabilità calcolata. L'interfaccia grafica consentirà all'utente di scegliere il numero di ripetizioni delle estrazioni simulate.

Si scriva un programma che disegni in una finestra un labirinto, rappresentando ad esempio le pareti con delle linee o dei blocchi. Il programma dovrà gestire un ipotetico visitatore del labirinto, consentendogli di muoversi autonomamente lungo il labirinto stesso (senza attraversare le pareti). Il visitatore potrà muoversi a caso o (opzionale) seguendo una particolare strategia. Il labirinto sarà costruito in maniera automatica in modo che sia diverso ad ogni avvio del programma.

Si scriva un programma che simuli il lancio di due dadi e visualizzi graficamente la frequenza dei risultati (somma dei valori sulla faccia superiore) ottenuti.

Si scriva un programma che gestisca liste d'esami universitari. Il programma dovrà consentire all'utente di selezionare un esame, identificato da un codice, dalla materia e dalla data di effettuazione, e di inserire il nominativo dello studente, a sua volta identificato da matricola, nome e cognome, che desideri iscriversi all'esame stesso. Il programma dovrà inoltre consentire di salvare su file e recuperare da file la situazione relativa alle iscrizioni. Dovrà inoltre permettere le più comuni operazioni di gestione, quali ad esempio la cancellazione e la modifica dei dati inseriti.

Si scriva un programma che definisca un calendario di appuntamenti. Ogni appuntamento è individuato da una data ed un'ora di inizio, da una data ed un'ora di fine e da una descrizione. Il programma, tramite un'opportuna interfaccia utente, consentirà di aggiungere ed eliminare appuntamenti, di scorrerli, nonché di visualizzare gli appuntamenti di una data giornata. Il programma consentirà inoltre di salvare l'elenco degli appuntamenti su un file e di leggere da file un elenco di appuntamenti precedentemente salvato. Aggiungere eventualmente al programma altre funzionalità, ad esempio la ricerca in base al campo descrizione.

Un numero intero è chiamato "perfetto" quando la somma dei suoi fattori, incluso 1 ma non se stesso, è pari a quel numero. Ad esempio 6 è un numero perfetto perché 6 = = 1 + 2 + 3. Si scriva un programma che visualizzi tutti i numeri perfetti, con i rispettivi fattori, inferiori od uguali ad un numero indicato dall'utente.

Si definisca una classe Paese, con dati di istanza il nome del paese, la sua popolazione e la sua area. Impiegando tale classe, si costruisca un programma che gestisca un insieme di paesi e consenta, selezionando un paese (ad esempio con una combobox), di visualizzarne i dati. Il programma dovrà inoltre visualizzare a richiesta i dati relativi al paese con l'area più grande, la popolazione più numerosa e la maggiore densità abitativa (persone per chilometro quadrato).

Si scriva un programma che consenta di inserire dei dati numerici e visualizzi un diagramma a torta ed un diagramma a barre dei dati inseriti.

Si realizzi un programma che simuli il gioco del lotto.
Il programma consentirà all'utente di simulare la puntata su uno o più numeri di una o più ruote.
Alla pressione di un pulsante il programma effettuerà l'estrazione casuale dei numeri sulle varie ruote e calcolerà l'eventuale vincita dell'utente.
Si inseriscano inoltre nel programma alcune semplici funzionalità statistiche.

NB: Le regole del gioco sono semplici e facilmente reperibili in Internet.

Si realizzi un programma che simuli il famoso gioco televisivo “Chi vuol essere milionario”.

Il gioco prevede che il concorrente (cioè, nel caso del programma, l’utente) risponda in sequenza ad una serie di domande, incrementando progressivamente il montepremi.
Per ogni domanda al concorrente vengono proposte quattro possibili risposte tra cui scegliere, una sola delle quali esatta.
Se il concorrente sceglie la risposta esatta, passa alla domanda successiva, e così in successione fino alla domanda finale con il montepremi più elevato. In caso di errore invece il gioco ha termine ed il concorrente è eliminato.

Il programma dovrà essere in grado di presentare le domande in modo casuale. Inoltre, le domande dovranno essere in qualche modo classificate in base alla difficoltà, in modo che a montepremi più elevati corrispondano domande più difficili.

Regole più dettagliate sul gioco, dalle quali trarre spunto per aggiungere ulteriori funzionalità al programma, sono facilmente reperibili in Internet.

Si realizzi un programma che consenta all’utente di digitare un testo e, alla pressione di un pulsante, visualizzi il numero totale ed in percentuale di ricorrenze di ogni lettera all’interno del testo stesso (maiuscole e minuscole vanno contate assieme).
Il programma inoltre dovrà determinare per ogni lettera del testo il numero di parole del testo stesso in cui ogni lettera è presente.

Si realizzi un programma che consenta all’utente di digitare un testo e, alla pressione di un pulsante, visualizzi (contando maiuscole e minuscole assieme):

- il numero totale e percentuale di ricorrenze di ogni lettera all’interno del testo stesso;
- il numero totale ed in percentuale di parole composte da 1, 2, 3, ecc. lettere;
- il numero totale di vocali e di consonanti presenti nel testo.

Si realizzi un programma che consenta di costruire piani di ammortamento.

Si realizzi un programma che consenta di calcolare il valore attuale ed il montante dei principali tipi di rendita finanziaria.

Si realizzi un programma che consenta all’utente di inserire le dimensioni di alcuni tipi di figure geometriche piane a scelta (ad esempio triangoli, rettangoli, pentagoni regolari, ecc.) o in alternativa di alcuni tipi di solidi (parallelepipedi, coni, ecc.) e che calcoli quantità come perimetro, area o volume.