Corso di Programmazione 1

Settima Esercitazione di Laboratorio

Osservazione preliminare: In tutti gli esercizi seguenti, suddividere il programma in due classi: `Programma` e `Metodi`. La classe `Programma` conterrà solamente il metodo main(), mentre la classe `Metodi` conterrà tutti gli altri metodi di supporto.

Esercizio 1 (Indovina il numero)

Scrivere un programma Java che consenta all'utente di giocare a "Indovina il numero".
Nel metodo main(), il programma estrae anzitutto a caso un numero intero compreso tra 1 e 100, invocando il metodo estraiNumeroCasuale() della classe `Metodi`. Entra poi in un ciclo, in cui chiede all'utente di indovinare il numero estratto a caso: chiede quindi di inserire da tastiera un numero intero compreso tra 1 e 100, e dice all'utente se il numero inserito è corretto, è più piccolo o è più grande di quello da indovinare.
Per rendere più intelligente il programma, memorizzare ad ogni iterazione:

- il numero *più piccolo*, tra quelli inseriti dall'utente, che sono *più grandi* del numero da indovinare;
- il numero *più grande*, tra quelli inseriti dall'utente, che sono *più piccoli* del numero da indovinare.

Così, se il numero da indovinare è 53, e l'utente inserisce i numeri 10, 80, 20, 83, il programma potrà rispondere come segue:

```
Il numero 10 e' piu' piccolo di quello da indovinare.
Il numero 80 e' piu' grande di quello da indovinare.
Il numero 20 e' piu' piccolo di quello da indovinare.
Hai inserito 83, ma se gia' ti avevo detto che il numero
   da indovinare e' piu' piccolo di 80…
```

Il confronto tra il numero inserito e quello da indovinare andrebbe fatto invocando il metodo confrontaNumeri(numeroInserito,numeroDaIndovinare), che restituisce −1, 0, 1 a seconda che il numero inserito sia minore, uguale o maggiore del numero da indovinare.
Inoltre:

- per ogni numero da indovinare, il giocatore ha a disposizione al più 10 tentativi;
- al termine della partita riporta le seguenti informazioni relative alle ultime 5 partite:
  - qual era il numero da indovinare
  - se l'utente l'ha indovinato oppure no
  - se l'utente ha indovinato il numero, con quanti tentativi lo ha fatto
- chiede poi all'utente se vuole giocare ancora, oppure se si vuole terminare il programma.

Suggerimento: le informazioni relative alle ultime 5 partite possono essere memorizzate in tre array: `numeriDaIndovinare` (di interi), `indovinato` (di boolean), e `numeroTentativi` (di interi). Gestire l'aggiornamento del contenuto di questi array tramite il metodo aggiornaInformazioniPartite().

Esercizio 2 (Indovina la parola)

Scrivere un programma Java che consenta all'utente di giocare a "Indovina la parola".
Nel metodo main(), il programma estrae anzitutto a caso una stringa da un array di stringhe prefissato, definito all'interno del metodo estraiStringaCasuale() della classe `Metodi` (il metodo restituisce al chiamante la stringa estratta). Converte poi la stringa in un array di `char`, ed entra quindi in un ciclo in cui:

- stampa le lettere della parola indovinate finora, e un trattino al posto di quelle non ancora indovinate. Ad esempio, se la parola segreta è `automobile`, e l'utente ha finora indovinato che la parola contiene delle `a`, delle `t` e delle `o`, il programma stampa:

  ```
  a-to-o----
  ```

  Inizialmente, quando l'utente non ha ancora indovinato nessuna lettera, il programma stamperà una sequenza di trattini, fornendo così al giocatore un'indicazione su quanto è lunga la parola da indovinare;
- chiede all'utente di inserire una lettera (per semplicità, supporre che le parole siano tutte formate solamente da lettere minuscole). Se la lettera non è presente nella parola, stampa un messaggio appropriato, e poi riesegue il ciclo (stampa lettere e trattini, chiede di inserire una lettera, ecc.). Similmente, se la lettera è già tra quelle indovinate allora stampa un messaggio appropriato e riesegue il ciclo.

Il gioco termina quando l'utente ha indovinato la parola, oppure quando ha fatto 20 tentativi.
Per la stampa delle lettere indovinate finora si invochi il metodo stampaLettereIndovinate(), passandogli gli opportuni argomenti. Per vedere se una lettera compare all'interno di una parola, e in tal caso rendere visibili tutte le occorrenze della lettera nella parola, invocare il metodo scopriLettera(), passandogli gli opportuni argomenti.

Esercizio 3 (Valutazione di espressioni aritmetiche semplici)

Scrivere un programma Java che, letta una semplice espressione aritmetica *da riga di comando*, ne calcoli il valore e lo stampi a video. Le espressioni aritmetiche riconosciute come valide hanno le seguenti caratteristiche:

- gli operatori sono solo + e – (considerati come operatori *binari*)
- gli operandi sono solo cifre comprese tra 0 e 9
- ogni espressione inizia e finisce con un operando
- operandi e operatori sono intercalati

Esempi di espressioni valide:

- 2 – 3 + 4
- 3
- 0 – 1 – 1 – 3

Esempi di espressioni non valide:

- 2 * 3 + 4 (non contiene solamente + e –)
- 12 + 4 (l'operando 12 non è una cifra compresa tra 0 e 9)
- – 3 (non inizia con un operando)
- 2 + – 5 (operandi e operatori non sono intercalati)

Come si diceva, l'espressione aritmetica viene specificata da riga di comando, quindi ad esempio:

```
java Programma 3 - 4 + 5
```

a cui il programma risponderà subito (senza ulteriori interazioni con l'utente): `4`.
Se l'espressione specificata non è valida, il programma stamperà a video un messaggio d'errore.
Il programma farà uso (almeno) dei seguenti metodi:

- controllaValiditaEspressione(): prende come argomento un riferimento all'array `args` dei parametri passati al programma da riga di comando, e verifica se l'espressione specificata in tali parametri è valida secondo le regole elencate sopra. Restituisce al chiamante un valore booleano;
- valutaOperando(): prende come argomento una stringa che rappresenta una cifra tra 0 e 9, e restituisce al chiamante il valore della cifra corrispondente (un `int`);
- valutaEspression(): prende come argomento un riferimento all'array `args` dei parametri passati al programma da riga di comando, e calcola il valore dell'espressione. Questo metodo va chiamato solo se il metodo controllaValiditaEspressione() ha precedentemente

restituito `true`, e fa uso del metodo valutaOperando() per convertire gli operandi da `String` a `int`.

Esercizio 4 (Trasposizione di una matrice)

Scrivere un programma Java che permetta all'utente di specificare il numero di righe e il numero di colonne (entrambe comprese tra 1 e 10) di una matrice casuale di elementi interi (compresi tra 0 e 99), la stampi a video e poi ne calcoli e stampi a video la *trasposta*.
Si ricorda che la trasposta di una matrice M è la matrice M^{T} ottenuta "scambiando" le righe con le colonne: cioè, le righe di M^{T} sono le colonne di M, e le colonne di M^{T} sono le righe di M.

Esempio:

$$M = \begin{bmatrix} 12 & 2 & 5 & 31 \\ 8 & 3 & 0 & 22 \\ 9 & 4 & 21 & 71 \end{bmatrix} \qquad M^{\mathrm{T}} = \begin{bmatrix} 12 & 8 & 9 \\ 2 & 3 & 4 \\ 5 & 0 & 21 \\ 31 & 22 & 71 \end{bmatrix}$$

Il programma deve usare i seguenti metodi:

- generaMatriceCasuale(), a cui si passa come argomenti il numero di righe e il numero di colonne della matrice da generare casualmente, e restituisce al chiamante un riferimento alla matrice generata;
- trasponiMatrice(), a cui si passa come argomento un riferimento alla matrice da trasporre, e restituisce al chiamante un riferimento alla matrice trasposta;
- stampaMatrice(), a cui si passa come argomento un riferimento alla matrice da stampare. Il metodo stampa la matrice a video e non restituisce nulla al chiamante.

Esercizio 5 (Cifrario di Vigenère)

Scrivere un programma Java che permetta all'utente di cifrare e decifrare dei messaggi usando il famoso *cifrario di Vigenère* (pubblicato nell'anno 1586). Il metodo main() invoca il metodo sceltaMenu(), che stampa a video il seguente menù e restituisce al chiamante la scelta fatta dall'utente:

```
1 - Inserisci/cambia la parola chiave
2 - Cifra un messaggio
3 - Decifra un messaggio
0 - Esci dal programma
```

Per semplicità supponiamo che i messaggi e le parole chiave siano formate solamente da lettere minuscole. Inizialmente la parola chiave non è ancora stata specificata, ed è quindi la stringa vuota. Tuttavia, non si possono cifrare o decifrare messaggi se prima non si è specificata una parola chiave; se l'utente prova a farlo, il programma deve stampare a video un messaggio d'errore.

Per *cifrare* un messaggio, il cifrario di Vigenère opera come segue.
Supponiamo che il messaggio da cifrare sia `programmazione`, e che la parola chiave sia `cane`. Ogni lettera del messaggio viene cifrata usando una lettera della parola chiave, e precisamente:

- la `p` di `programmazione` con la `c` di `cane`
- la `r` di `programmazione` con la `a` di `cane`
- la `o` di `programmazione` con la `n` di `cane`
- la `g` di `programmazione` con la `e` di `cane`
- la `r` di `programmazione` con la `c` di `cane`
- la `a` di `programmazione` con la `a` di `cane`
- la `m` di `programmazione` con la `n` di `cane`
- … e così via, finché non è stato cifrato tutto il messaggio.

In pratica, le lettere della parola chiave vengono usate a rotazione, finché non è terminato il messaggio da cifrare.

Per cifrare una lettera del messaggio (supponiamo `p`) usando una lettera della parola chiave (supponiamo `c`), si usa la seguente matrice: in particolare, si incrociano la riga che inizia con `p` e la colonna che inizia per `c`, e la lettera contenuta nella casella così individuata (`r`) costituisce la cifratura di `p` con la chiave `c`.

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*	*X*	*Y*	*Z*
B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

La cifratura del messaggio `programmazione` con la parola chiave `cane` risulta quindi essere:

Lettere del messaggio: `programmazione`
Lettere della parola chiave: `canecanecaneca`
Lettere del messaggio cifrato: `rrbktazqczvspe`

Per *decifrare* il messaggio `rrbktazqczvspe` con la parola chiave `cane`, ancora una volta si usano i caratteri della parola chiave, a rotazione:

Lettere del messaggio cifrato: `rrbktazqczvspe`
Lettere della parola chiave: `canecanecaneca`
Lettere del messaggio decifrato: `programmazione`

Per decifrare una lettera del messaggio cifrato (supponiamo `r`) usando una lettera della parola chiave (supponiamo `c`), si usa ancora la matrice di cui sopra. In particolare, nella colonna che inizia per `c` si cerca il carattere `r` da decifrare; il carattere decifrato è quello che si trova all'inizio della riga corrispondente (`p`).

Il programma dovrà utilizzare i seguenti metodi:

- cambiaParolaChiave(), che chiede all'utente di specificare una nuova parola chiave, controlla che sia formata da caratteri minuscoli e la restituisce al chiamante;
- creaMatrice(), che crea la matrice riportata sopra e ne restituisce un riferimento al chiamante;
- cifraLettera(), a cui si passa come argomenti il carattere da cifrare e il carattere della parola chiave da usare, e restituisce al chiamante il carattere cifrato;
- decifraLettera(), a cui si passa come argomenti il carattere da decifrare e il carattere della parola chiave da usare, e restituisce al chiamante il carattere decifrato;
- cifraMessaggio(), a cui si passa come argomenti il messaggio da cifrare e la parola chiave, e restituisce al chiamante il messaggio cifrato;

- decifraMessaggio(), a cui si passa come argomenti il messaggio da decifrare e la parola chiave, e restituisce al chiamante il messaggio decifrato.

Il programma dovrà consentire all'utente di cifrare e decifrare messaggi, e cambiare la parola chiave, fino a che selezionerà la voce "`0 - Esci dal programma`" nel menù.