Esercitazioni di Laboratorio

6. Laboratorio 6

Come sempre a fine laboratorio compilate il questionario su wooclap

Esercizio 0

Presentazioni dei Tutor Junior:

- 5) ALTRE REGOLE PER L'IMPAGINAZIONE DEL CODICE
- 6) ESERCIZIO SVOLTO: ZIA BETTINA E LE IMMAGINI DISTORTE (Esercizio 10 del Lab 5)

Esercizio 1

Argomento: array, cicli

Realizzare una classe MyFirstArray che contenga un metodo main in cui venga creato un array di interi di dimensione 10. Inizializzare ciascuna posizione dell'array con il valore dell'indice corrispondente (ad esempio, a[2]=2);

Parte a)

Stampare a video la lunghezza dell'array.

Stampare a video il contenuto di ciascuna cella a partire dalla cella in posizione 0 fino ad arrivare all'ultima.

Stampare a video il contenuto di ciascuna cella a partire dalla cella in ultima posizione fino ad arrivare alla prima.

Parte b)

Realizzare un metodo ausiliario statico con firma

public static void incrementAll(int[] a)

che prenda come parametro esplicito un array di interi e che lo modifichi al suo interno incrementando il valore di ogni posizione di una unita'. Il tipo di dati restituito dal metodo e' void.

Dopo aver invocato il metodo dal main, passando come parametro l'array creato precedentemente , stampare a video il contenuto di ciascuna cella dell'array a partire dalla cella in posizione 0. Il contenuto e' stato modificato?

Parte c)

Creare nel main una variabile int e assegnarle un valore.

Realizzare un metodo ausiliario statico con firma

public static void incrementVar(int value)

che prenda come parametro esplicito una variabile intera e che la modifichi al suo interno incrementando il valore di una unita'. Il tipo di dati restituito dal metodo e' void.

Stampare a video il contenuto della variabile prima e dopo l'invocazione dal main del metodo incrementVar() a cui venga passato come parametro la variabile precedentemente creata. Il contenuto e' stato modificato?

Se ti torna la risposta alla domanda precedente guarda questa simpatica gif.

Esercizio 2

Argomento: lettura di parametri da linea di comando, array, cicli

Realizzare una classe ComLine in cui si inseriscano dei parametri da linea di comando, ad esempio, in esecuzione: java ComLine 3 ciao 48

Se non vengono inseriti parametri riportare a standard output il messaggio: "Non sono stati inseriti parametri", altrimenti elencare, in ordine, uno sotto l'altro, i parametri inseriti.

Esercizio 3

Realizzare la classe ArrayUtil che contiene al suo interno il metodo resize visto in classe, con l'aggiunta della possibilita' che il valore del parametro esplicito newLength sia minore di quella dell'array. In questo caso restituire un array con lo stesso contenuto (fino alla nuova dimensione) e troncato alla dimensione indicata.

Realizzare una versione di resize che ridimensiona un array di int e una che ridimensiona un array di double. Il nome del metodo puo' rimanere invariato. Questa proprieta' si chiama "sovraccarico". Il compilatore e' in grado di distinguere tra le due in base al tipo di parametro passato.

Esercizio 4

Argomento: gestione dinamica di un array, cicli

Leggere un numero arbitrario di interi da standard input e riportarli in uscita in ordine inverso di inserimento.

Riferimento Soluzione: InvertArrayOrder.java

Esercizio 5

Argomento: array, cicli

Chiedere all'utente di inserire un numero intero. Leggere due serie di numeri in virgola mobile di lunghezza pari all'intero specificato. Ciacun numero della serie deve essere specificato in una riga di input (= premere invio dopo ogni numero inserito). Restituire in uscita la somma di ciascuna componente corrispondente e la somma totale. Ad esempio:

Quanti elementi?

4

Inserisci gli elementi del primo array, uno per riga:

1.2

2.3

3.4

4.5

Inserisci gli elementi del secondo array, uno per riga:

1.1

0

2.2

2.2

In output:

2.3 2.3 5.6 6.7

16.9

Esercizio 6

Argomento: array riempiti solo in parte, semplici algoritmi su array

Aggiungere alla classe ArrayUtil gli algoritmi su array visti a lezione utilizzando array riempiti solo in parte. Ricordarsi che, poiche' le variabili

primitive passati come argomenti sono passate per valore e non modificabili, e' necessario modificare la dimensione logica nel metodo chiamante (nel caso di questo esercizio nel metodo main della classe ArrayUtilTester).

- generazione di un array di interi di numeri casuali (randomIntArray);

public static int[] randomIntArray(int length, int n)

- stampare il contenuto di un array (printArray);

public static String printArray(int[] v, int vSize)

- eliminare un elemento (ordine non importante);

public static void remove(int[] v, int vSize, int index)

- eliminare un elemento (ordine importante)

public static void removeSorted(int[] v, int vSize, int index)

- inserire un elemento

public static int[] insert(int[] v, int vSize, int index, int value)

- ricerca del minimo

public static int findMin(int[] v, int vSize)

- ricerca del massimo

public static int findMin(int[] v, int vSize)

- ricerca di un valore in modo sequenziale

public static int find(int[] v, int vSize, int target)

Testare gli algoritmi scrivendo una classe di collaudo **ArrayUtilTester** che

- riceva da standard input la dimensione e l'intervallo di variabilità di un array di numeri interi casuali
- Stampi il contenuto dell'array generato
- Collaudi gli algoritmi visualizzando dopo ogni operazione il contenuto dell'array attraverso la sequenza di richieste:

1. Chiedere all'utente un valore e una posizione in cui inserirlo
2. Chiedere all'utente di indicare la posizione di un elemento da eliminare (senza considerare rilevante l'ordine)
3. Chiedere all'utente di indicare la posizione di un elemento da eliminare (considerando rilevante l'ordine)
4. Chiedere all'utente un valore da cercare
5. Visualizzi minimo e massimo del contenuto dell'array

Ricordarsi di modificare la taglia di vSize se le operazioni che la modificano vanno a buon fine.

Esercizio 7

Argomento: gestione dinamica di un array unidimensionale, cicli, array bidimensionali

Scrivere il programma IsMagicSquare che verifichi se un quadrato di numeri è un "quadrato magico".

Una disposizione bidimensionale di numeri con dimensione n x n è un quadrato magico se contiene i numeri interi da 1 a n^2 e se la somma degli elementi di ogni riga, di ogni colonna e delle due diagonali principali ha lo stesso valore. Esempi di quadrati magici, con n=4 e n=5:

```
16   3   2  13
 5  10  11   8
 9   6   7  12
 4  15  14   1
```

```
11  18  25   2   9
10  12  19  21   3
 4   6  13  20  22
23   5   7  14  16
17  24   1   8  15
```

L'utente introduce i numeri del (presunto) quadrato in sequenza, uno o più numeri per riga, senza alcuna relazione con il numero di righe del quadrato: quando l'utente ha introdotto tutti i numeri e ha chiuso lo standard input, il programma deve intraprendere le azioni seguenti:

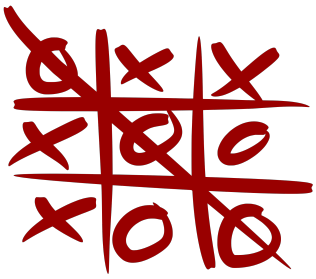
- verifica che il numero di valori introdotti sia il quadrato di un numero intero n: in caso contrario, il programma termina segnalando un fallimento;
- verifica che la sequenza di valori introdotta contenga tutti (e soli) i numeri da 1 a n^2, senza ripetizioni: in caso contrario, il programma termina segnalando un fallimento (pensate bene a come implementare questo controllo, si puo' fare, anzi si deve fare, solo con quanto visto finora);
- dispone i valori all'interno di un array bidimensionale, riempito per righe seguendo l'ordine con cui sono stati forniti i valori: il primo valore prende posto nell'angolo in alto a sinistra, il secondo nella seconda posizione da sinistra della prima riga e così via, riempiendo righe successive del quadrato;
- verifica la validità delle regole del quadrato magico, interrompendo la verifica con la segnalazione di fallimento non appena una regola non sia rispettata;
- segnala il successo della verifica.

Esempio di input per il primo quadrato:

```
16 3 2 13 5 10
11 8 9 6 7 12 4
15
14 1
```

Esercizio 8

Argomento: progettazione di classi, uso di array bidimensionali



Scrivere un programma per giocare a "tris" (tic-tac-toe in inglese), il classico gioco in cui due giocatori dispongono alternativamente un proprio contrassegno in una casella di una scacchiera 3 x 3 finché uno dei due non pone tre contrassegni in una fila orizzontale, verticale o diagonale.

X	O	
	X	O
		X

Il programma inizia visualizzando la scacchiera vuota, come segue (ogni puntino rappresenta una casella vuota)

```
|...|
|...|
|...|
```

e chiedendo al primo giocatore di inserire le coordinate della casella in cui vuole porre il suo contrassegno (che sara` un carattere X).
Le coordinate si indicano con due numeri interi (valori ammessi: 0, 1 o 2), il primo numero essendo l'indice di riga (a partire dall'alto) ed il secondo l'indice di colonna (a partire da sinistra).
Il programma deve verificare se la casella richiesta è libera oppure no. Nel primo caso visualizza la scacchiera aggiornata e chiede all'altro giocatore di inserire la propria mossa (verrà usato il contrassegno O). Nel secondo caso, invece, il programma fornisce una segnalazione d'errore, visualizza nuovamente la stessa scacchiera visualizzata in precedenza e chiede al giocatore di inserire una nuova mossa; analogo comportamento si verifica se il giocatore introduce delle coordinate non valide, ma il messaggio d'errore deve essere diverso.
Il programma deve essere in grado di segnalare la vittoria di uno dei due giocatori qualora questa avvenga, oppure di porre fine alla partita quando la scacchiera è piena senza che uno dei due giocatori abbia raggiunto la vittoria.
Al termine di una partita, il programma chiede al giocatore se intende giocare un'altra partita oppure no: nel secondo caso il programma termina.

Risolvere il problema in due passi

1. Scrivere una classe **MyTris** che rappresenti la scacchiera e la cui interfaccia pubblica è definita a questo link (css)
2. Scrivere poi una classe eseguibile che usi **MyTris**, e che gestisca una partita a tris tra due giocatori, seguendo il comportamento descritto sopra.

Esercizio 9

Argomento: progettazione di classi

Implementare la classe Player che rappresenta un giocatore e la classe eseguibile Risiko che simula un turno al noto gioco.

In particolare, la classe Player dovra' tener conto del nome del giocatore,
dei tre lanci di dado (facciamo che siano sempre tre) e del punteggio (numero di tiri vincenti) . Nella classe dovranno essere inoltre definiti:

```
// costruttore: il punteggio iniziale e' 0 cosi' come
// i valori dei tiri dei tre dadi
public Player(String aName)

// metodi di accesso
public String getName();
public String getScore();

// simula un turno di lancio di dadi attribuendo a ciascun
// lancio un valore casuale tra 1 e 6
public void turno()

// restituisce un riferimento a un nuovo array di interi in
// cui i valori ottenuti nei tre lanci sono ordinati
public int[] sortDice()

//aggiorna il punteggio incrementandolo di una unita'
public void addPoint()

//resetta il punteggio
public void resetScore()

//restituisce una stringa con il nome del giocatore e
//i valori dei lanci dei dadi
public String toString()
```

Il metodo turno() della classe Player fa uso della classe java.util.Random che rende disponibili metodi per generare numeri pseudo-casuali, ovvero quasi casuali. Consultare l'interfaccia pubblica della classe Random, in particolare il metodo int nextInt(int n).

Scrivere una classe eseguibile Risiko.java che effettua le seguenti operazioni:
- acquisisce da standard input il nome del primo giocatore
- acquisisce da standard input il nome del secondo giocatore
- definisce un oggetto di classe Player per ciascuno dei due giocatori
- lancia tre volte il dado per ciascun giocatore (si usa il metodo turno())
- stampare i risultati di ciascun giocatore (stampare implicitamente con toString())
- dichiari un riferimento ad un array di interi e gli assegni il risultato dell'invocazione del metodo sortDice() per il primo giocatore
- stampare i valori ordinati dei lanci seguite dal nome del giocatore
- ripetere le due operazioni precedenti per il secondo giocatore
- confrontore i risultati ottenuti nei lanci dai due giocatori assegnando un punto (chiamata al metodo addScore()) al giocatore che vince ciascun confronto
-verificare chi sia il vincitore e stampare in uscita il nome e i punteggi
- invia a standard output il risultato dei lanci
- calcola il vincitore e invia il risultato a standard output (metodo String vincitore(...)).

In esecuzione la classe Risiko dovra' produrre un risultato simile al seguente:

$ java Risiko
Giocatore n. 1? Pippo
Giocatore n. 2? Pluto
lanci di Pippo: 2 3 1
lanci di Pluto: 6 6 2
lanci ordinati
1 2 3 Pippo
2 6 6 Pluto
Pluto vince 3 a 0

NB: in caso di pareggio vince il secondo giocatore (che si difende).