Esercizi per il Corso di Metodologie di Programmazione

Capitolo 7

Università degli Studi di Roma "La Sapienza"

Esercizio 1

Creare un array di numeri interi con una dimensione pari a 10 e assegnare a ciascuna cella un valore letto da input. Successivamente, stampare il contenuto dell'array partendo dall'ultimo elemento e andando a ritroso.
Esempio: contenuto nel file *example_1.txt*.

Esercizio 2

Creare un array di numeri interi con una dimensione pari a 10 e assegnare a ciascuna cella un valore aleatorio tra 1 e 100, estremi inclusi. Successivamente, il programma dovrà:

- Stampare tutti gli elementi di indice pari
- Stampare tutti gli elementi con un valore pari
- Stampare tutti gli elementi in ordine inverso
- Stampare gli elementi in prima e ultima posizione

Esempio: contenuto nel file *example_2.txt*
Consiglio: utilizzare il metodo nextInt() della classe *Random* per produrre dei valori interi aleatori.

Esercizio 3

Creare un array di numeri *double* con una dimensione pari a 100 e assegnare a ciascuna cella un valore inserito dall'utente. L'inserimento potrà essere interrotto prima del raggiungimento della dimensione massima qualora l'utente inserisca un valore non numerico. Successivamente, il programma dovrà stampare ciascun elemento dell'array indicando quale sia il suo numero massimo e minimo.
Esempio: contenuto nel file *example_3.txt*

Esercizio 4

Creare un array di numeri *double* con una dimensione pari a 100 e assegnare a ciascuna cella un valore inserito dall'utente. L'inserimento potrà essere interrotto prima del raggiungimento della dimensione massima qualora l'utente inserisca un valore non numerico. Successivamente, il programma dovrà stampare la somma di tutti gli elementi dell'array a meno del minimo.
Esempio: contenuto nel file *example_4.txt*

Esercizio 5

Creare una classe chiamata *Student* con la seguente struttura:

- **Attributi:**
  - *scores*: array di valori *double* che contiene i voti inseriti
  - *scoresSize*: valore di tipo intero che contiene il numero di voti inseriti
- **Costruttore:** accetta un parametro di tipo intero che indica la capacità dell'array con la quale verrà inizializzato. All'attributo *scoresSize* verrà assegnato un valore pari a 0.
- **Metodi:**
  - *addScore*: funzione di tipo *boolean* che aggiunge un voto, passato come parametro, all'interno dell'array. Se l'inserimento ha avuto successo, la funzione ritornerà il valore *true*. Nel caso in cui sia stata raggiunta la capacità massima invece, la funzione ritornerà il valore *false*
  - *sum*: ritorna la somma dei valori contenuti nell'array
  - *minimum*: ritorna il valore minimo contenuto nell'array
  - *finalScore*: ritorna la somma degli elementi a meno del minimo. Se l'array non contiene elementi, la funzione ritornerà un valore pari a 0. In caso di un unico elemento presente, la funzione ritornerà tale valore
  - *removeMin*: modifica l'array eliminando il valore minimo e riducendone la dimensione

  Per testare il corretto funzionamento, creare la classe *StudentDemo* la quale, al suo interno, istanzierà un oggetto di tipo *Student* con un numero massimo di voti pari a 100. Successivamente, il programma aggiungerà una sequenza di voti inserita dall'utente, terminata da un carattere non numerico, gestendo il caso in cui sia stato raggiunto il valore massimo di elementi. Ad array valorizzato, il programma rimuoverà il valore minimo al suo interno e stamperà il punteggio totale senza contare ulteriormente il valore minimo.
  Esempio: contenuto nel file *example_5.txt*

Esercizio 6

Creare una classe chiamata *DataSet* con la seguente struttura:

- **Attributi:**

  - *data*: array di valori *double*
  - *dataSize*: valore di tipo intero che contiene il numero di elementi inseriti
- **Costruttore:** inizializza l'array *data* con una dimensione pari a 100, mentre a *dataSize* verrà assegnato un valore pari a 0.
- **Metodi:**
  - *add*: aggiunge un voto, passato come parametro, all'interno dell'array. In caso di raggiungimento del numero massimo di valori inseribili, l'array raddoppierà la sua dimensione, mantenendo tutti i dati contenuti in precedenza
  - *alternatingSum*: produce un valore che è pari alla differenza tra i numeri in posizione pari e dispari contenuti all'interno dell'array. In caso di array vuoto la funzione ritornerà un valore pari a 0

Per testare il corretto funzionamento, creare la classe *DatasetDemo* la quale, al suo interno, istanzierà un oggetto di tipo *Dataset* che conterrà una sequenza di numeri inserita dall'utente, terminata da un carattere non numerico. Ad array valorizzato, il programma stamperà il risultato di *alternatingSum*.
Esempio: contenuto nel file *example_6.java*
Consiglio: per raddoppiare la dimensione di un array, utilizzare il metodo *copyOf* della classe Arrays.

Esercizio 7

Creare una classe chiamata *ReverseElements* che contiene il metodo *reverse*. Tale metodo accetterà come parametro un array di valori *double* e ritornerà il suo contenuto in senso opposto. Testarne successivamente il corretto funzionamento inserendo una sequenza di numeri.
Esempio: contenuto nel file *example_7.txt*.

Esercizio 8

Creare una classe chiamata *PermutationGenerator* con la seguente struttura:

- **Attributi:**
  - *generator*: di tipo Random, è il generatore di numeri utili alla permutazione
  - *length*: valore di tipo intero che indica il numero di elementi da generare all'interno della permutazione
- **Costruttore:** inizializza l'oggetto di tipo *Random* con un parametro il cui valore è pari a 42, mentre a *length* verrà assegnato un valore pari al parametro del costruttore.
- **Metodi:**
  - *nextPermutation*: produce una permutazione di numeri che vanno da 1 al valore inserito nel costruttore

Esempio: per testare il corretto funzionamento della classe, utilizzare *example_8.java*. All'interno della classe verranno generate e stampate diverse permutazioni a fronte di un'istanziazione con un numero di elementi pari a 10 e 5.
Consiglio: qui potrete trovare le informazioni necessarie per comprendere il significato di permutazione.

Esercizio 9

Creare una classe chiamata *Swap* che contiene il metodo *swapHalves*. Tale metodo accetterà come parametro un array di valori interi e ritornerà una sua versione modificata in cui le due metà dell'array saranno scambiate tra loro. Testarne successivamente il corretto funzionamento inserendo una sequenza di numeri.
Esempio: contenuto nel file *example_9.txt*.

Esercizio 10

Creare una classe chiamata *DataSet* con la seguente struttura:

- **Attributi:**
  - *data*: array di valori *double*
  - *dataSize*: valore di tipo intero che contiene il numero di elementi inseriti
- **Costruttore:** inizializza l'array *data* con una dimensione pari a quella fornita come parametro, mentre a *dataSize* verrà assegnato un valore pari a 0.
- **Metodi:**
  - *add*: aggiunge un valore, passato come parametro, all'interno dell'array. In caso di raggiungimento del numero massimo di valori inseribili non sarà possibile inserire altro
  - *getSum*: ritorna la somma dei valori all'interno dell'array
  - *getAverage*: ritorna il valore medio dei valori all'interno dell'array
  - *getMaximum*: ritorna il valore massimo dell'array. In caso di array vuoto, la funzione ritornerà il più piccolo valore *double* possibile
  - *getMinimum*: ritorna il valore minimo dell'array. In caso di array vuoto, la funzione ritornerà il più grande valore *double* possibile

Esempio: per testare il corretto funzionamento della classe, utilizzare *example_10.java*.

Esercizio 11

Dato un array di interi di dimensione pari a 10 e valorizzato dall'utente attraverso una sequenza di input, stampare a video un messaggio che indichi se all'interno di tale array siano presenti o meno dei duplicati.
Esempio: contenuto nel file *example_11.txt*.

Esercizio 12

Creare una classe chiamata *Sequence* con la seguente struttura:

- **Attributi:**
  - *values*: array di valori interi
- **Costruttore:** inizializza l'array *values* con una dimensione pari a quella fornita come parametro.
- **Metodi:**
  - *set*: riceve in input due parametri che indicano la posizione in cui il nuovo elemento dovrà essere inserito all'interno dell'array della classe insieme all'elemento stesso. Il metodo dovrà quindi assegnare alla cella nella posizione desiderata il valore indicato
  - *get*: ritorna l'elemento nella posizione indicata come parametro del metodo
  - *size*: ritorna il numero di valori presenti nella sequenza
  - *equals*: metodo di tipo *boolean* che confronta la sequenza con un'altra passata come parametro. Il metodo ritornerà un valore pari a *false* se le due sequenze sono diverse oppure se ciascun elemento dei due array nella stessa posizione differisce nel valore. In caso contrario, il metodo ritornerà un valore pari a *true*

Esempio: per testare il corretto funzionamento della classe, utilizzare *example_12.java*.

Esercizio 13

Estendere l'esercizio precedente aggiungendo il metodo *sameValues* che prende in input un oggetto di tipo *Sequence* e controlla che ciascun elemento di ogni array abbia una corrispondenza nell'altro. Se la corrispondenza è mantenuta da entrambi, il metodo ritornerà un valore *true*. In caso contrario, il metodo ritornerà *false*.
Esempio: per testare il corretto funzionamento della classe, utilizzare *example_13.java*.

Esercizio 14

Estendere l'esercizio precedente aggiungendo il metodo *sum* che prende in input un oggetto di tipo *Sequence* ed effettua la somma di entrambe le sequenze con la seguente logica:

- Se l'indice *i* di una cella è presente in entrambe le sequenze, il risultato della somma comprenderà il valore di entrambi gli elementi in quella posizione
- Se l'indice *i* di una cella è maggiore della lunghezza di una delle due sequenze, il risultato della somma comprenderà solamente il valore dell'elemento presente in quella posizione

Esempio: per testare il corretto funzionamento della classe, utilizzare *example_14.java*.

Esercizio 15

Dato un array di dimensione pari a 10 e valorizzato con valori aleatori tra 1 e 100, stampare a video il secondo numero più grande al suo interno.
Esempio: contenuto nel file *example_15.txt*.

Esercizio 16

Dato un array di interi con una dimensione pari a 10, valorizzato con valori aleatori tra 1 e 100, stampare a video un messaggio che indichi se il contenuto dell'array segua un ordine (crescente o decrescente) o meno.
Esempio: contenuto nel file *example_16.txt*.

Esercizio 17

Dato un array di interi con una dimensione pari a 10, valorizzato con valori aleatori tra 1 e 100, modificare il suo contenuto in modo tale che ciascun valore venga sostituito con il valore massimo dei suoi vicini, eccetto l'ultimo. Successivamente, stampare a video il contenuto del nuovo array.
Esempio: contenuto nel file *example_17.txt*.

Esercizio 18

Creare una classe chiamata *Table* con la seguente struttura:

- **Attributi:**
  - *values*: matrice di valori interi
- **Costruttore:** inizializza la matrice *values* le cui dimensioni sono pari a quelle fornite dai parametri.
- **Metodi:**
  - *set*: riceve in input tre parametri che indicano rispettivamente la riga, la colonna ed il valore da inserire nella posizione indicata dai dalle coordinate precedenti
  - *sum*: effettua la somma della riga o della colonna in base al valore (booleano) del primo parametro, la cui posizione è indicata dal secondo. *Esempio:* la chiamata *sum(true, 2)* farà sì che vengano sommati tutti gli elementi in riga 2. Al contrario, chiamare il metodo *sum(false, 1)* porterà ad ottenere la somma di tutti gli elementi in colonna 1.

Esempio: per testare il corretto funzionamento della classe, utilizzare *example_18.java*.

Esercizio 19

Dato un array di interi con una dimensione pari a 10, valorizzato con valori aleatori tra 1 e 100, modificare il suo contenuto in modo tale che ciascun valore pari venga sostituito uno 0.

Successivamente, stampare a video il contenuto del nuovo array.
Esempio: contenuto nel file *example_19.txt*.

Esercizio 20

Creare una classe chiamata *BarChart* con la seguente struttura:

- **Attributi:** nessuno
- **Costruttore:** di default
- **Metodi:**
  - *readValues*: legge da input una sequenza di numeri interi positivi, terminata da un valore negativo. Ciascun valore verrà aggiunto come elemento all'interno di una lista di tipo *ArrayList<Integer>* il quale, ad inserimento terminato, verrà ritornata come output della funzione
  - *findMax*: data una lista di tipo *ArrayList<Integer>* passata come parametro alla funzione, viene individuato il valore massimo al suo interno, ritornandolo come output
  - *printBarChart*: riceve come parametro una lista di tipo *ArrayList<Integer>* e stampa ciascun valore sotto forma di grafico a barre formato da asterischi. La scala con cui rappresentare ciascuna unità, ossia il numero di asterischi, è ottenuta dividendo il valore 20 per il numero massimo all'interno della lista. Ad esempio, se la lista contiene i valori 5, 5, 2, la scala sarà pari a 4 in quanto risultato dell'operazione 20/5, dove 5 è il valore massimo trovato al suo interno. Questo vuol dire che ciascuna unità sarà rappresentata da 5 asterischi

Ai fini dell'esercizio dovrà essere creato un oggetto di tipo *BarChart*, riproponendo l'esempio contenuto nel file *example_20.txt*.