Java

Eserciziario

R-DeV
Revolution Development
learning@r-dev.it

Esercizio 1 - gestione dei cicli

A. Scrivere una funzione che stampi a video gli interi pari compresi fra 50 e 0, partendo da 50.
B. Scrivere una funzione che stampi la tabellina del numero dato come argomento
C. Scrivere un programma che stampi i primi 15 elementi della successione di Fibonacci.

Esercizio 2 - gestione del flusso

Scrivere un metodo che calcoli il risultato dell'esame di uno studente. Il programma prenderà in ingresso:

- Il voto ottenuto nella prova di teoria (variabile tra -8 e +8)
- Il voto ottenuto nella prova pratica (variabile tra 0 e 24)

Memorizzati questi dati procederà al calcolo del risultato finale in trentesimi procedendo come segue:

- Il risultato finale è la somma dei risultati.
- se il voto di teoria è minore (o uguale) di zero e la somma dei voti di teoria e pratica è maggiore di 18 lo studente è bocciato.
- se il voto di teoria è minore (o uguale) di zero e il voto di pratica è minore di 18 lo studente è bocciato.
- se il voto di teoria è maggiore di zero e la somma dei voti di teoria e pratica è minore di 18 lo studente è bocciato.
- se la somma di teoria e pratica vale 31 o 32 il programma stampa: "congratulazioni: 30 e lode"
- in tutti gli altri casi lo studente è promosso e viene riportato il voto calcolato.

Esercizio 3 - stringhe

A. Scrivere una funzione `stampaVocali` che, data una stringa, ne stampa le sole vocali
B. Scrivere una funzione che stampi al contrario una stringa presa in input
C. Scrivere una funzione `contaChar(char c, String s)` che restituisce il numero delle occorrenze di c in s
D. Scrivere una funzione `isAnagramma(String s1, String s2)` che restituisce true se le stringhe in input sono una l'anagramma dell'altra
E. Scrivere una funzione `isPalindroma(String s)` che restituisce true se s è palindromo

Esercizio 4 - array

A. Scrivere una funzione di *utilities* che riempie un array di interi random
B. Scrivere una funzione che restituisce il massimo in un array di interi
C. Scrivere una funzione `boolean contains(int e, int[] array)` che restituisce true se *`array`* contiene `e`, false altrimenti
D. Scrivere una funzione che restituisce la media degli elementi di un array di interi
E. Scrivere il metodo: `public int mostRecurrent(int [] array)` che trova l'elemento più ricorrente in un array. Il metodo restituisce l'elemento trovato
F. Scrivere un metodo che calcoli la media di un array di numeri interi, considerando i soli numeri divisibili per tre.

Esercizio 5.A - equals

Data la seguente classe.

```
public class Z {
     private Z other;
     private int val;
     ...
}
```

Si considerino le seguenti specifiche alternative per il metodo equals: Due oggetti x e y di tipo Z sono uguali se:

1) x.other e y.other puntano allo stesso oggetto ed x.val è maggiore o uguale di y.val
2) x.other e y.other puntano allo stesso oggetto ed x.val e y.val sono entrambi pari
3) x.other e y.other puntano allo stesso oggetto oppure x.val è uguale a y.val
4) x.other e y.other sono entrambi null oppure nessuno dei due è null ed x.other.val è uguale a y.other.val

- Dire quali specifiche sono valide e perché
- Implementare la specifica 4

Esercizio 5.B - equals

Realizzare la classe Engine, che rappresenta un motore a combustione, caratterizzato da cilindrata (in cm3) e potenza (in cavalli). Normalmente, due oggetti Engine sono uguali se hanno la stessa cilindrata e la stessa potenza. Il metodo `byVolume` converte questo Engine in modo che venga confrontata solo la cilindrata. Analogamente, il metodo `byPower` converte questo Engine in modo che venga confrontata solo la potenza

```
Engine a = new Engine(1200, 69), b = new Engine(1200, 75), c =
new Engine(1400, 75);
System.out.println(a);
System.out.println(a.equals(b));
Engine aVol = a.byVolume(), bVol = b.byVolume();
System.out.println(aVol);
System.out.println(aVol.equals(bVol));
System.out.println(a==aVol);
Engine bPow = b.byPower(), cPow = c.byPower();
System.out.println(bPow);
System.out.println(bPow.equals(cPow))
```

Risultato:

```
(1200.0 cm3, 69.0 CV)
false
(1200.0 cm3, 69.0 CV)
true
false
(1200.0 cm3, 75.0 CV)
true
```

Esercizio 5.C - equals

Implementare la classe Studente e le due sottoclassi Triennale e Magistrale. Uno studente è caratterizzato da nome e matricola.

Ciascuna sottoclasse ha un prefisso che viene aggiunto a tutte le sue matricole.

Due studenti sono considerati uguali da equals se hanno lo stesso nome e la stessa matricola (compreso il prefisso).

L'implementazione deve rispettare il seguente caso d'uso:

```
Studente.Triennale.setPrefisso ("N86");
Studente.Magistrale.setPrefisso ("N97");
Object luca1 = new Studente.Triennale("Luca", "004312"),
luca2 = new Studente.Triennale("Luca", "004312"),
anna1 = new Studente.Triennale("Anna", "004312"),
anna2 = new Studente.Magistrale("Anna", "004312");
System.out.println(luca1.equals(luca2));
System.out.println(anna1.equals(anna2));
System.out.println(anna1);
```

Risultato:

```
true false Anna N86/004312
```

Esercizio 6 - classi interne

Un oggetto Curriculum rappresenta una sequenza di lavori, ognuno dei quali è un'istanza della classe Job. Il costruttore di Curriculum accetta il nome di una persona. Il metodo addJob aggiunge un lavoro alla sequenza, caratterizzato da una descrizione e dall'anno di inizio, restituendo un nuovo oggetto di tipo Job. Infine, la classe Job offre il metodo next, che restituisce in tempo costante il lavoro successivo nella sequenza (oppure null).

Implementare le classi Curriculum e Job, rispettando il seguente caso d'uso:

```
Curriculum cv = new Curriculum("Walter White");
Curriculum.Job j1 = cv.addJob("Chimico", 1995);
Curriculum.Job j2 = cv.addJob("Insegnante", 2005);
Curriculum.Job j3 = cv.addJob("Cuoco", 2009);
System.out.println(j1.next());
System.out.println(j2.next());
System.out.println(j3.next());
```

Risultato:

```
Insegnante: 2005
Cuoco: 2009
null
```

Esercizio 7.A - enumerazioni

Realizzare la classe enumerata Note, che rappresenta le sette note musicali. Le note sono disposte su una scala di frequenze, in modo che ciascuna nota sia separata dalla successiva da due semitoni, tranne il MI, che è separato dal FA da un solo semitono.

La classe ScalaDiFrequenza ha il metodo getInterval, che accetta due oggetti Note e restituisce il numero di semitoni tra le due note. Si faccia in modo che il metodo getInterval funzioni in tempo costante, cioè indipendente dall'argomento che gli viene passato.

```
System.out.println(ScalaDiFrequenze.getInterval(Note.DO, Note.MI));
System.out.println(ScalaDiFrequenze.getInterval(Note.MI, Note.LA));
System.out.println(ScalaDiFrequenze.getInterval(Note.LA, Note.SOL));
```

Risultato:

```
4
5
-2
```

Esercizio 7.B - enumerazioni

L'enumerazione Nutrient contiene i valori GRASSI, CARBOIDRATI e PROTEINE. Implementare la classe NutrInfo che rappresenta la scheda nutrizionale di un prodotto gastronomico. Il suo costruttore accetta il nome del prodotto e la quantità. Il metodo setNutrient permette di impostare la quantità (in grammi) di ciascun nutriente. L'implementazione deve rispettare il seguente caso d'uso:

```
NutrInfo x = new NutrInfo("pane pizza",500);
x.setNutrient(Nutrient.GRASSI, 12.0);
x.setNutrient(Nutrient.CARBOIDRATI, 20.0);
x.setNutrient(Nutrient.PROTEINE, 15.0);
System.out.println(x);
```

Risultato:

```
Valori nutrizionali per 500.0 grammi di pane pizza:
proteine:15.0
grassi:12.0
carboidrati:20.0
```

Esercizio 8.A - ereditarietà e polimorfismo

Si creino le classi Prodotto, Televisore e Smartphone. Un televisore è caratterizzato da marca, modello, prezzo, numero di pollici e un booleano che indica se è smartTv o meno; uno smartphone è caratterizzato da marca, modello, prezzo, sistema operativo, anno di produzione e gb di ram. Oggi è black friday e il venditore applica lo sconto del 20% a tutte le tv e del 30 % a tutti gli smartphone.

Rispettare il seguente caso d'uso:

```
Prodotto tvLg = new Televisore("LG", "nuovo", 1000, 48, true);
Prodotto umidigi = new Smartphone("Umidigi", "F2", 250, "Android 10.0", 6,
"2020");
tvLg.applicaSconto();
umidigi.applicaSconto();
System.out.println(tvLg.getPrezzo());
System.out.println(umidigi.getPrezzo());
```

Risultato

```
800.0

175.0
```

Esercizio 8.B - ereditarietà e polimorfismo

Aggiungiamo una funzionalità all'esercizio precedente (8.A).

- Creiamo la classe Carrello dove inseriremo i prodotti acquistati dal cliente.
- Ogni cliente può acquistare massimo 10 prodotti. Se si tenta di acquistare l'11 prodotto verrà stampata a video un messaggio di errore
- Se il totale del carrello supera i 2500 euro il venditore applica un ulteriore sconto del 5%
- l'inserimento di un nuovo prodotto deve avvenire in tempo costante
- il metodo acquista applica gli sconti previsti dal black friday

Caso di test:

```
Prodotto tvLg = new Televisore("LG", "AXC", 1000, 48, true);
Prodotto umidigi = new Smartphone("Umidigi", "F2", 250, "Android 10.0", 6, "2020");
Prodotto tvSony = new Televisore("Sony", "WDE", 3000, 56, true);
Prodotto samsung = new Smartphone("Samsung", "S7", 300, "Android 6.0", 2, "2017");
Prodotto akai = new Televisore("Akai", "BGT", 235, 32, false);
Carrello c = new Carrello();
c.addProdotto(tvLg);
c.addProdotto(tvSony);
c.addProdotto(samsung);
c.addProdotto(umidigi);
c.addProdotto(akai);
System.out.println("totale = " + c.acquista());
```

Risultato: `totale = 3584.35`

Esercizio 8.C - ereditarietà e polimorfismo

Gestione di un garage seguendo le seguenti specifiche:

- il garage ha massimo 15 posti, ogni posto è identificato da un progressivo (da 0 a 14)
- il garage può ospitare solo auto, moto e furgoni
- implementare una funzione che inserisce un nuovo veicolo nel garage nel primo posto disponibile effettuando gli opportuni controlli
- implementare una funzione che inserisce un nuovo veicolo in un determinato posto preso in input effettuando gli opportuni controlli
- implementare una funzione che estrae un veicolo nel garage prendendo in input il numero del posto e ritornando in output l'istanza del veicolo stesso
- implementare una funzione che stampi la situazione corrente dei posti nel garage
- implementare in ogni classe il metodo toString()

Esercizio BO - Generics

Scrivere un metodo statico getMedian che accetta un array qualsiasi e restituisce l'elemento mediano (di posto intermedio)