Attività di laboratorio

Laboratorio 4

Esercizio 1

Argomento: decisioni; confronto tra stringhe

Scrivere un programma che

- chiede all'utente di inserire tre stringhe (una per riga)
- visualizza le stringhe in ordine lessicografico crescente (una per riga)

Esercizio 2

Argomento: decisioni; confronto tra numeri reali

Scrivere un programma che

- legge due numeri in virgola mobile
- visualizza un messaggio che dice se i due numeri sono o meno *approssimativamente* uguali

I due numeri sono considerati *approssimativamente* uguali se, quando vengono arrotondati con due cifre decimali dopo la virgola, differiscono per meno di **0.01**.
Esempio: i due numeri **2.0** e **1.99998** vengono considerati approssimativamente uguali. I due numeri **0.999** e **0.991** non vengono considerati approssimativamente uguali.

Esercizio 3

Argomento: decisioni e operatori booleani

Scrivere un programma che segnala all'utente se il numero intero positivo che ha introdotto corrisponde ad un anno bisestile oppure no

Esercizio di verifica
Kaltura Media Gallery
I miei corsi

AMMINISTRAZIONE
Gestione libro
Stampa libro
Stampa questo capitolo
Amministrazione del corso

COMMUNITY FINDER
Cerca

Suggerimento: un anno bisestile è divisibile per 4. Fanno eccezione gli anni divisibili per 100, che non sono bisestili, e gli anni divisibili per 400, che invece sono bisestili: tali eccezioni esistono però solo dopo l'adozione del calendario gregoriano, che avvenne nel 1582.

Esercizio 4

Argomento: decisioni e operatori booleani

Scrivere un programma **SimpleTriangoloTester**, che riceve da standard input tre numeri interi positivi che rappresentano le lunghezze dei lati di un triangolo e che invia a standard output una stringa contenente le seguenti informazioni:

- relativamente ai lati: equilatero, isoscele, scaleno
- relativamente agli angoli: acutangolo, rettangolo, ottusangolo

Esempi

- se vengono inseriti i numeri "3 4 5", il programma visualizzera` la stringa "triangolo scaleno rettangolo".
- se vengono inseriti i numeri "5 7 7", il programma visualizzera` la stringa "triangolo isoscele acutangolo".
- se vengono inseriti i numeri "5 3 3", il programma visualizzera` la stringa "triangolo isoscele ottusangolo".
- se vengono inseriti i numeri "3 3 3", il programma visualizzera` la stringa "triangolo equilatero" (non serve l'informazione relativa agli angoli perche' i triangoli equilateri sono sempre acutangoli avendo tre angoli uguali pari a pigreco / 3).
- se vengono inseriti i numeri "3 4 8", il programma visualizzera` la stringa "non è un triangolo" (non sempre tre lati rappresentano un triangolo).

Alcuni suggerimenti (da non leggere subito ma da consultare solo in caso di difficolta`) sono disponibili a questo link.

Esercizio 5

Argomento: decisioni e operatori booleani

Scrivere un programma che riceve da standard input la data di compleanno dell'utente (in formato giorno mese), e visualizza a standard output l'oroscopo corrispondente. Il programma deve gestire correttamente anche il caso in cui l'input non corrisponda al formato prescritto.
Esempio: Se viene inserito l'input "26 7" (ovvero, 26 luglio), il programma potra` visualizzare la seguente stringa

```
LEONE
Amore:    4/5
Amicizia: 3/5
Lavoro:   3/5
```

Se invece l'input inserito non e` interpretabile come una data in formato giorno mese (ad esempio, l'input "43 21", oppure l'input "32 1", oppure l'input "30 2"), il programma dovra` visualizzare la stringa

```
L'input inserito non e` una data.
```

Suggerimenti: le associazioni tra date di nascita e segni zodiacali possono essere reperite in rete, ad esempio qui. Le stringhe contenenti gli oroscopi di ciascun segno possono venire inventate a vostro piacimento (come si fa solitamente con gli oroscopi...), l'unico requisito è che ciascuna di esse deve contenere il nome del segno corrispondente.

Attenzione: gli esercizi seguenti hanno come argomento la progettazione ed il collaudo di classi. E` bene seguire le seguenti fasi:

1. Progettare la classe
   - Stabilire quali sono le caratteristiche essenziali degli oggetti della classe, e fare un elenco delle operazioni che sara` possibile compiere su di essi: processo di *astrazione*
   - Definire e scrivere l'*interfaccia pubblica*. Ovvero, scrivere l'intestazione della classe, definire i costruttori ed i metodi pubblici da realizzare, e scrivere la *firma* (specificatore di accesso, tipo di valore restituito, nome del metodo, eventuali parametri espliciti) di ciascuno di essi.
     Consiglio: se un metodo non restituisce valori (ovvero il tipo del valore restituito e` void), scrivete inizialmente un corpo *vuoto*, ovvero **{}**. Se un metodo restituisce valori non void, scrivete inizialmente un corpo fittizio contenente solo un enunciato di return: ad esempio **{return 0;}** per metodi che restituiscono valori numerici o char, **{return false;}** per metodi che restituiscono valori booleani, **{return null;}** per metodi che restituiscono riferimenti ad oggetti. Con questi accorgimenti il codice compilera` correttamente fin dall'inizio del vostro lavoro. Quando poi scriverete i metodi, modificherete le istruzioni di return secondo quanto richiesto da ciascun metodo.
   - Definire le *variabili di esemplare* ed eventuali *variabili statiche*. E` necessario individuare tutte le variabili necessarie (quante? quali? dipende dalla classe!). Per ciascuna di esse si deve, poi, definire tipo e nome.
2. Realizzare la classe
   - Verificate le impostazioni del vostro editor di testi, al fine rendere piu` efficiente il vostro lavoro di programmazione. In particolare, verificate ed eventualmente modificate le impostazioni per i rientri di tabulazione (valori tipici sono di 3 o 4 caratteri), e visualizzate i numeri di riga.
   - Scrivere il codice dei metodi.
     Consiglio: non appena si e` realizzato un metodo, si deve compilare e correggere gli errori di compilazione (se il corpo del metodo e` particolarmente lungo e complicato, compilare anche prima di terminare il metodo). Non aspettate di aver codificato tutta la classe per compilare! Altrimenti vi troverete, molto probabilmente, a dover affrontare un numero elevato di errori, con il rischio di non riuscire a venirne a capo in un tempo ragionevole (come quello a disposizione per la prova d'esame...).
3. Collaudare la classe. Ovvero scrivere una *classe di collaudo* contenente un metodo main, all'interno del quale vengono definiti e manipolati oggetti appartenenti alla classe da collaudare.
   - E` possibile scrivere ciascuna classe in un file diverso. In tal caso, ciascun file avra` il nome della rispettiva classe ed avra` l'estensione .java. Tutti i file vanno tenuti nella stessa cartella, tutti i file vanno compilati separatamente, solo la classe di collaudo (contenente il metodo main) va eseguita.
   - E` possibile scrivere tutte le classi in un unico file. In tal caso, il file .java deve contenere una sola classe public. In particolare, la classe contenente il metodo main deve essere public mentre la classe (o le classi) da collaudare non deve essere public (non serve scrivere private, semplicemente non si indica l'attributo public). Il file .java deve avere il nome della classe public

Esercizio 6

Argomento: progettazione e collaudo di classi

Progettare una versione modificata della classe **BankAccount** vista a lezione. L'interfaccia pubblica della versione modificata della classe deve contenere, oltre ai metodi precedentemente definiti e realizzati, anche il seguente metodo:

```
public void addInterest(double rate)
```

Questo metodo aggiunge al saldo del conto gli interessi, calcolati al tasso percentuale **rate** specificato come parametro esplicito. Il metodo deve essere realizzato in maniera tale che il parametro esplicito **rate** non possa essere un valore negativo. Ad esempio, dopo l'esecuzione di questi enunciati

```
BankAccount account = new BankAccount(2000);
account.addInterest(2);   //interessi al 2%
```

il saldo di **account** è di 2040€.
Inoltre il costruttore **BankAccount(double initialBalance)** e i metodi **deposit(double amount)** e **withdraw(double amount)** devono essere realizzati in maniera tale che

- il loro parametro esplicito (**initialBalance** o **amount**) non possa essere un valore negativo
- il saldo del conto corrente non possa diventare negativo dopo un prelievo.

Progettate una versione modificata della classe **BankAccountTester** vista a lezione, che

- riceva da input standard due numeri in virgola mobile, che specificano rispettivamente il saldo iniziale e il tasso percentuale di interesse
- crei un conto bancario accreditando un saldo iniziale ricevuto da standard input come numero in virgola mobile
- usi il metodo **addInterest** per accreditare sul conto bancario gli interessi di due anni, calcolati su un tasso di interesse ricevuto da standard input come numero in virgola mobile
- visualizzi il saldo del conto dopo due anni
- effettui un prelievo dal conto, pari ad una cifra ricevuta da standard input come numero in virgola mobile
- visualizzi il saldo del conto dopo il prelievo.

Verificare in particolare il funzionamento del programma quando vengono inseriti valori non validi (saldo iniziale negativo, tasso di interesse negativo, prelievo negativo o superiore al saldo).

Esercizio 7

Argomento: progettazione e collaudo di classi

Realizzare una classe **Product** (ovvero, prodotto).
Un prodotto è caratterizzato da un nome e da un prezzo in €. La classe **Product** deve contenere il costruttore

```
public Product(String name, double price)
```

che crea un nuovo prodotto caratterizzato dal nome **name** e dal prezzo in euro **price**. La classe deve inoltre contenere i seguenti metodi pubblici:

```
public String getName()
public double getPrice()
public void reducePrice(double rate)
```

I primi due sono *metodi di accesso*, e restituiscono rispettivamente il nome e il prezzo del prodotto. Il terzo e` un *metodo modificatore*, che riduce il prezzo del prodotto secondo lo sconto percentuale definito dal parametro esplicito **rate**.
Scrivere una classe di collaudo che

- crea due prodotti ricevendo i parametri di creazione da standard input
- stampa i due prodotti (ovvero il loro nome e prezzo) a standard output, in ordine decrescente di prezzo
- applica uno sconto al prodotto più caro, secondo una percentuale ricevuta da standard input
- stampa nuovamente i due prodotti (ovvero il loro nome e prezzo) a standard output, in ordine di prezzo

---

Esercizio 8 (impegnativo)

Argomento: progettazione e collaudo di classi

Si scriva la classe **Cerchio**, che descrive un cerchio, la cui interfaccia pubblica è specificata a questo link.
Un cerchio e` caratterizzato da due dati: il centro e il raggio. A sua volta il centro, essendo un punto di un piano, e` caratterizzato da due dati: ascissa e ordinata.
Suggerimento: dati due cerchi, siano ***O=(x,y)*** e ***R*** il centro e il raggio del cerchio ***C***, e ***O'=(x',y')*** e ***R'<R*** il centro e il raggio del cerchio ***C'***. Allora

- **C'** e` interno a **C** se **OO'<R-R'**
- **C'** e` tangente interno a **C** se **OO' = R-R'**
- **C'** e` secante a **C** se **R-R'<OO'<R+R'**
- **C'** e` tangente esterno a **C** se **OO' = R+R'**
- **C'** e` esterno a **C** se **OO'>R+R'**

Collaudare la classe **Cerchio** con la classe di prova **CerchioTester**, che legge i dati da standard input, e che è disponibile a questo link.