Fondamenti di Architetture e Programmazione
III Compitino

Esercizi di preparazione

Cognome ..... Nome .....
Matricola .....

- Usate il retro dei fogli per scrivere lo svolgimento degli esercizi di programmazione.
- Nei seguenti quesiti, quando vi è richiesto di scrivere un programma, potete limitarvi al *corpo* del metodo main, assumendo se necessario che in e out siano due variabili di classe ConsoleInputManager e ConsoleOutputManager (rispettivamente), già dichiarate e inizializzate.

1. Scrivete un programma che legga un numero n, seguito da una sequenza di esattamente n stringhe; al termine, deve ristampare ciascuna stringa, un carattere sì e uno no.

   Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

   ```
   Quante stringhe:  4
   Stringa 1:  mamma
   Stringa 2:  pipo
   Stringa 3:  xyp231
   Stringa 4:  urka
   mma
   pp
   xp3
   uk
   ```

(Svolgimento sul retro)

2. Considerate la classe Cibo i cui oggetti corrispondono a dei *cibi*; questa classe ha un costruttore pubblico

   - public Cibo(String nome, int kcalEtto)

   che costruisce un cibo denominato nome, che ha un apporto calorico all'etto pari a kcalEtto. La classe possiede i metodi

   - public String getNome()
   - public int kcal(int grammi)
   - public int maxEttiAlGiorno()

   che forniscono, rispettivamente, il nome del cibo, il numero complessivo di calorie per un dato numero di grammi, e la massima quantità giornaliera consentita, supponendo che ci si nutra solamente di questo cibo e non si vogliano superare le 2000 kcalorie complessive.

   Scrivete un programma che operi come segue:

   (a) chieda all'utente di inserire un numero intero, diciamo n;

   (b) dichiari un array di n cibi e lo riempia con cibi inseriti dall'utente;

   (c) stampi i soli cibi dei quali si possa mangiare più di 1 Kg. al giorno, indicando per ciascuno il numero di chilogrammi che si possono mangiare di quel cibo in un giorno (la classe Cibo dispone di un opportuno metodo toString()).

   Ecco un esempio di esecuzione (le parti in grassetto sono state inserite dall'utente):

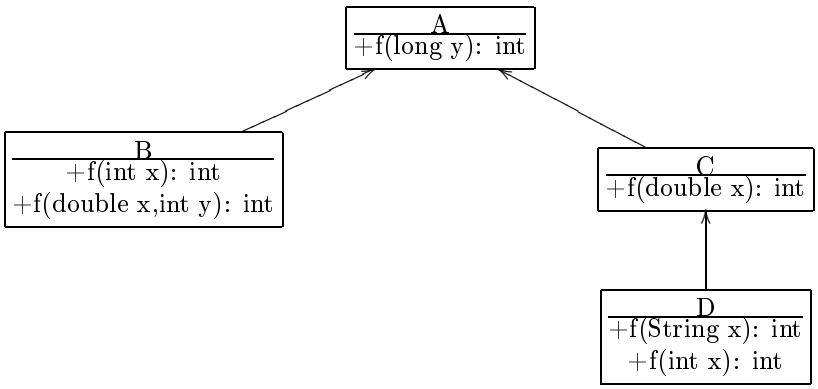
   Inserisci un intero: **4**

   Nome: **Tonno sott'olio**
   KCal/h: **258**
   Nome: **Coniglio**
   KCal/h: **150**
   Nome: **Lattuga**
   KCal/h: **20**
   Nome: **Maionese**
   KCal/h: **655**

   Coniglio Kg. 1.333333
   Lattuga Kg. 10.0

(Svolgimento sul retro)

3. Considerate la seguente gerarchia dei tipi (rappresentata mediante un diagramma UML):



e ipotizzate che tutte le classi abbiano un costruttore pubblico senza argomenti. Assumete le seguenti definizioni e inizializzazioni:

```
A a1, a2, a3, a4;
B b;
C c1, c2;
D d;

a1 = new A(); a2 = new B(); a3 = new C(); a4 = new D();
b = new B();
c1 = new C(); c2 = new D();
D d = new D();
```

Per ciascuno dei seguenti assegnamenti, dite se l'assegnamento è consentito, oppure se richiede (per poter essere compilato) un operatore di cast nel secondo membro (e in tal caso indicate l'assegnamento corretto), oppure se l'assegnamento non è consentito nemmeno con un cast (nel senso che anche con un cast produrrebbe un errore di compilazione o un'eccezione in esecuzione):

- b=a2: .....
- b=a1: .....
- a1=c1: .....
- a2=b: .....
- a1=c1: .....
- c2=c1: .....
- c1=(C)a2: .....
- c1=a3: .....

- c1=(C)a3: .....

4. Facendo riferimento al precedente diagramma UML e alle variabili definite e inizializzate come sopra, considerate le seguenti invocazioni di metodo. Per ciascuna di esse dovete dire se è consentita e in caso affermativo dovete indicare il nome della classe che contiene il metodo che verrà effettivamente eseguito:

   - d.f("pippo"): .....
   - d.f(3,4): .....
   - a1.f(3,4): .....
   - a3.f(5): .....
   - a4.f("pippo"): .....
   - b.f(3): .....
   - d.f(3.4): .....
   - a2.f(3,4): .....

5. La classe Integer contiene un metodo statico

```
public static String toBinaryString( int x )
```

che dato un intero x lo converte in una stringa binaria. Avete una variabile intera i che contiene un numero intero e volete mettere nella variabile booleana b il valore true se e solo se la lunghezza della rappresentazione binaria del quadrato del contenuto di i è maggiore di 5; scrivete l'istruzione di assegnamento a b:

6. Considerate la seguente funzione definita ricorsivamente sugli interi:

$$f(x) = \begin{cases} 2 * x & \text{se } x \text{ è pari} \\ x + f(x+1) & \text{altrimenti} \end{cases}$$

Scrivete un metodo statico con intestazione

```
public static int f( int x )
```

per il calcolo di f.

7. Scrivete una classe di nome Prodotto le cui istanze rappresentano i prodotti in vendita in un supermercato. Ogni prodotto è caratterizzato da un nome (p.es. "Pannoloni Morbidoni Fluffy"), un prezzo unitario espresso da un double (p.es. 3.12), e un numero intero che rappresenta quantità disponibile (p.es. 1500).

   La classe deve contenere i seguenti metodi e costruttori:

   - Prodotto(String nome, **double** prezzo, **int** disp): crea un prodotto con dato nome, prezzo unitario e quantità disponibile;
   - Prodotto(String nome, **double** prezzo): come sopra, ma assume che la quantità disponibile sia zero;
   - **void** acquista(**int** qta): aumenta la quantità disponibile del valore qta;
   - **void** vendi(**int** qta) **throws** IllegalStateException: riduce la quantità disponibile del valore qta, e solleva un'eccezione se la quantità disponibile è inferiore (l'eccezione IllegalStateException è un'eccezione già esistente, con gli usuali costruttori);
   - **double** prezzoTotale(**int** qta): restituisce il prezzo totale relativo alla quantità qta;
   - **boolean** scorteFinite(): restituisce true se le scorte di questa merce sono terminate (cioè, se la quantità disponibile è zero);
   - String toString(): restituisce una stringa descrittiva.

(Svolgimento sulla facciata seguente)

8. Definite una classe di nome ProdottoScontato che estende Prodotto e i cui oggetti corrispondono a dei prodotti con sconto. Questa classe deve avere:

   - un costruttore pubblico

     ```
     ProdottoScontato(String nome, double prezzo, int disp, double sconto)
     ```

     che specifica il nome del prodotto, il suo prezzo unitario, la quantità disponibile e la percentuale di sconto (p.es., se sconto vale 0.3 significa che il prodotto viene venduto con il 30% di sconto);

   - un metodo

     ```
     public boolean scontoAlto()
     ```

     che restituisce true se e solo se il prodotto ha uno sconto maggiore del 50%;

   - il metodo ereditato prezzoTotale(**int** qta) deve restituire il prezzo al netto dello sconto.