Esercizio 1.

Il package `prog.utili` fornisce una classe `Importo`, i cui oggetti rappresentano importi in euro. Tale classe mette a disposizione i seguenti costruttori:

Constructor and Description
`Importo(int x)` Costruisce un nuovo oggetto che rappresenta un importo in cui il numero di Euro, senza centesimi, è specificato dall'argomento.
`Importo(int x, int y)` Costruisce un nuovo oggetto che rappresenta un importo in cui il numero di Euro è specificato dal primo argomento e il numero di centesimi di Euro dal secondo argomento.

I metodi rilevanti sono i seguenti:

Modifier and Type	Method and Description
`int`	**`compareTo(Importo x)`** Confronta l'importo che esegue il metodo con quello specificato come argomento.
`int`	**`diviso(Importo x)`** Restituisce il quoziente della divisione dell'importo che esegue il metodo per l'importo fornito come argomento.
`Importo`	**`diviso(int x)`** Restituisce il riferimento a un nuovo oggetto che rappresenta l'importo ottenuto dividendo l'importo rappresentato dall'oggetto che esegue il metodo per l'intero fornito tramite l'argomento.
`boolean`	**`equals(Importo altro)`** Restituisce `true` se l'importo rappresentato dall'oggetto che esegue il metodo è uguale a quello specificato tramite l'argomento, `false` in caso contrario.
`boolean`	**`equals(java.lang.Object o)`** Restituisce `true` se e solo se l'oggetto specificato come argomento è un'istanza di `Importo` e l'importo che esegue il metodo è uguale a quello specificato tramite l'argomento.
`static Importo`	**`fromLire(int x)`** Restituisce il riferimento a un nuovo oggetto che rappresenta l'importo in Euro equivalente al numero di lire fornito tramite l'argomento.
`int`	**`getCent()`** Restituisce il numero di centesimi dell'importo rappresentato dall'oggetto che esegue il metodo.
`int`	**`getEuro()`** Restituisce il numero di Euro (senza centesimi) dell'importo rappresentato dall'oggetto che esegue il metodo.
`int`	**`hashCode()`**
`boolean`	**`isMaggiore(Importo x)`** Restituisce `true` se l'importo rappresentato dall'oggetto che esegue il metodo è maggiore di quello specificato tramite l'argomento, `false` in caso contrario.

boolean	**isMinore**(**Importo** x) Restituisce `true` se l'importo rappresentato dall'oggetto che esegue il metodo è minore di quello specificato tramite l'argomento, `false` in caso contrario.
Importo	**meno**(**Importo** x) Restituisce il riferimento a un nuovo oggetto che rappresenta l'importo ottenuto sottraendo l'importo fornito tramite l'argomento da quello rappresentato dall'oggetto che esegue il metodo.
Importo	**per**(int x) Restituisce il riferimento a un nuovo oggetto che rappresenta l'importo ottenuto moltiplicando l'importo rappresentato dall'oggetto che esegue il metodo per l'intero fornito tramite l'argomento.
Importo	**piu**(**Importo** x) Restituisce il riferimento a un nuovo oggetto che rappresenta l'importo ottenuto sommando l'importo fornito tramite l'argomento a quello rappresentato dall'oggetto che esegue il metodo.
int	**toLire**() Restituisce il valore in lire corrispondente all'importo rappresentato dall'oggetto che esegue il metodo
java.lang.String	**toString**() Restituisce il riferimento a una stringa che rappresenta il valore dell'importo che esegue il metodo (la stringa contiene nelle due posizioni più a destra le cifre corrispondenti ai centesimi, anche se queste sono zero).

Scrivete un'applicazione che legga i prezzi dei prodotti acquistati e la percentuale di sconto, e comunichi il prezzo totale da pagare, avendo applicato lo sconto solo prodotto più costoso.

Esempio di output:

```
Prezzo: euro? 4
cent? 33
Altri prodotti? s
Prezzo: euro? 10
cent? 60
Altri prodotti? s
Prezzo: euro? 5
cent? 12
Altri prodotti? n
Sconto? 5
TOTALE EURO 20,05
SCONTO 5% Su 10,60: EURO 0,5
TOTALE SCONTATO: EURO 19,52
```

Esercizio 2.

Il package `java.util` fornisce una classe `Random` le cui istanze sono generatori di numeri con distribuzione casuale. La classe dispone di un costruttore privo di argomenti che crea un generatore. Tra i metodi forniti vi sono:

```
public int nextInt()
```

```
public boolean nextBoolean()
```

che restituiscono, rispettivamente, un valore `int` e un valore `boolean` generati a caso.

```
public int nextInt(int n)
```

che restituisce un valore intero, generato a caso, maggiore o uguale a zero e minore del valore specificato tramite l'argomento.

```
public int nextDouble()
```

che restituisce un valore `double` generato a caso nell'intervallo [0,1[.

Utilizzando il metodo `nextBoolean` della classe Random, scrivete un'applicazione che simuli una sequenza di lanci di moneta, dove il numero di lanci viene inserito preliminarmente dall'utente. Calcolate via via la percentuale di lanci che forniscono come risultato "testa" e la percentuale di lanci che forniscono come risultato "croce". Al crescere del numero di lanci le due percentuali dovrebbero stabilizzarsi a intorno al 50%.

Esercizio 3.

La classe `Math` del package `java.lang` fornisce un metodo statico `log` che riceve come argomento un valore double e ne restituisce il logaritmo naturale (sempre rappresentato come double). Scrivete un'applicazione che riceva in ingresso due numeri in virgola mobile *x* e *y* (con y >= x >= 1) e calcoli (in maniera approssimata) l'integrale da *x* a *y* della funzione logaritmo.

Il calcolo può basarsi sulla seguente idea (nell'ipotesi che y >= x >= 1):

1. Si prenda in esame un rettangolo con base coincidente con l'intervallo considerato, cioè da *x* a *y*, e con altezza uguale al valore massimo della funzione nell'intervallo (in questo caso *log*(*y*)).
2. Si generano molti punti a caso all'interno di questo rettangolo.
3. Il rapporto tra il numero di punti che si trovano al di sotto della funzione e il numero di punti generati sarà circa uguale al rapporto tra l'integrale da calcolare e l'area del rettangolo.

Implementate quest'idea nella vostra applicazione, utilizzando la classe `Random`, descritta nell'Esercizio precedente. Algoritmi che utilizzano generatori casuali per il calcolo, prendono anche il nome di "metodi Monte Carlo".