Da una prova scritta di POO

Una classe **Denaro** è associata ad una specifica moneta o banconota in euro. Attributi di un oggetto Denaro sono: il suo *valore* (es. 10.00 per una banconota da dieci euro) e la sua *quantità*, ossia il numero dei pezzi disponibili di quel valore. I possibili valori sono quelli correntemente in circolazione ovvero: 0.01, 0.02, 0.05, 0.10, 0.20, 0.50, 1.00, 2.00, 5.00, 10.00, 20.00, 50.00, 100.0, 200.00 e 500.00. Se si tenta di creare un oggetto Denaro di un taglio differente occorre sollevare un'opportuna eccezione. La classe Denaro implementa l'interfaccia **Comparable** con il criterio che d1 precede d2 se il valore di d1 è maggiore di quello di d2. La classe Denaro espone (almeno) i metodi: costruttori, getValore(), getQuantita(), setQuantita(...), equals(), toString(), hashCode().

Un'interfaccia **Soldi** modella una somma di denaro in euro rappresentata da una collezione ordinata di oggetti Denaro. L'interfaccia estende **Comparable** e **Iterable** ed esporta i seguenti metodi:

- double **totale**(), che ritorna il valore complessivo della somma di denaro;
- void **add**(Denaro d), che aggiunge alla somma di denaro (this) il contenuto dell'oggetto d ricevuto come parametro;
- void **add**(Soldi s), che aggiunge alla somma di denaro this la somma di denaro s ricevuta come parametro;
- Soldi **sub**(Soldi s1, Soldi s2), che effettua un "pagamento" s1 con una somma s2. In particolare, il metodo aggiunge alla somma di denaro this la somma s2 e restituisce un oggetto Soldi contenente la differenza (resto) "s2-s1" prelevandola dalla somma this. Un resto va considerato "esatto" a meno di una differenza di al più tre centesimi di euro. Se non è possibile fornire il resto, il metodo deve sollevare un'opportuna eccezione e lasciare this inalterato. Si nota ancora che l'oggetto Soldi da restituire deve essere composto dal minimo numero di oggetti Denaro sulla base di quelli disponibili in this. Il metodo solleva un'ulteriore eccezione se il totale di s2 è più piccolo del totale di s1.

Concretizzare mediante metodi default, quanti più metodi è possibile nell'interfaccia **Soldi**. Quindi sviluppare una classe astratta **SoldiAbstract** che implementa l'interfaccia **Soldi** e fornisce i metodi equals(), hashCode() e toString().
Sviluppare quindi una classe concreta **SoldiLC** erede di **SoldiAbstract** che memorizza la collezione di oggetti **Danaro** in una lista concatenata.

Fornire, infine, una classe **Applicazione** col main, che simula la cassa di un supermercato. Su un file testo fcassa.txt è memorizzato il contenuto iniziale della cassa (ogni linea contiene una moneta/banconota e la sua quantità). Da input si leggono quindi l'ammontare di una spesa e una somma di denaro da utilizzare per il pagamento. Il programma effettua il pagamento e visualizza il risultato dell'operazione.