

# PROGRAMMAZIONE II (A,B) - a.a. 2013-14

## Prima Valutazione Intermedia — 31 marzo 2014

1. Una `CoinsCollection` è una collezione di monete del valore di 50 centesimi, 1 o 2 euro. Un valore tipico è

$$[v1*50C, v2*1E, v3*2E]$$

dove  $v1$ ,  $v2$  e  $v3$  sono valori interi maggiori di zero.

2. Fornire (scrivendo un'interfaccia Java) la specifica del tipo di dati astratto modificabile `CoinsCollection` con le seguenti operazioni:

- `createCC()` crea una `CoinsCollection` vuota;
- `addCoin(Integer coin)` inserisce la moneta di valore `coin` nella collezione;
- `balance()` restituisce il valore totale della collezione di monete;
- `makeChange(Integer amount)` effettua l'operazione di "cambio" del valore `amount` in monete presenti nella collezione: restituisce, se possibile, una `CoinsCollection` con un valore totale uguale a `amount`, togliendo le monete corrispondenti da `this`.

Fornire, oltre all'istituzione dei metodi, l'overview (basata sul punto 1.), le clausole `REQUIRES`, `MODIFIES` e `EFFECTS` di ogni metodo, indicando anche le eccezioni eventualmente lanciate e se sono `checked` o `unchecked`.

3. Supponendo di utilizzare come struttura di rappresentazione del TDA `CoinsCollection` una lista di `Integer` della forma

```
private List<Integer> monete;  
// lista della monete presenti nella collezione
```

definire la funzione di astrazione e l'invariante di rappresentazione.

4. Fornire l'implementazione del metodo `balance`.  
Fornire l'implementazione del metodo `makeChange` e dimostrare che preserva l'invariante di rappresentazione.
5. Supponiamo di aggiungere al TDA `CoinsCollection` il metodo `getCoins` così definito:

```
public List<Integer> getCoins() { return new ArrayList<Integer>(monete); }
```

Sapendo che il costruttore `ArrayList<Integer>(Collection <Integer>)` restituisce una nuova lista in cui vengono copiati tutti gli oggetti della collezione passata come parametro, dire se l'introduzione di questo metodo determina l'esposizione della rappresentazione della classe, cioè se questo metodo può essere usato da un programmatore per falsificare l'invariante di rappresentazione di `CoinsCollection`.

6. Un ragionevole test per il metodo `makeChange` è costituito dalla creazione di una `CoinsCollection`, seguita dall'aggiunta di un certo numero di monete con `addCoin`, e da una o più invocazioni di `makeChange` con opportuni argomenti, seguite da controlli sulla collezione restituita o sull'eccezione eventualmente lanciata.

Definire una batteria di test per il metodo `makeChange`, cioè un insieme di test sufficiente a convincersi che il metodo implementato soddisfa la specifica.

7. Si definisca il TDA `FloppyCoinsCollection` esattamente come `CoinsCollection`, tranne che per il metodo `makeChange` che viene definito come segue:

- `makeChange(Integer amount)` restituisce una `FloppyCoinsCollection` con un valore totale uguale a `val`, dove `val` è il più piccolo tra `amount` e `this.balance()`.

Dire, motivando la risposta, se `FloppyCoinsCollection` è un sottotipo di `CoinsCollection`, cioè se è soddisfatto il principio di sostituzione.