```java
import java.util.ArrayList;

/**
 * Classe che simula un borsellino
 * @author radaelli11353
 */
public class Purse {
    private ArrayList<String> coins;

    /**
     * Costruttore
     * @param dim Quantità massima di monete inseribili nel borsellino
     */
    public Purse(int dim) {
        coins = new ArrayList<String>(dim);
    }

    /**
     * Metodo che aggiunge una moneta al borsellino
     * @param coinName Nome della moneta
     */
    public void addCoin(String coinName) {
        coins.add(coinName);
    }

    /**
     * Metodo che inverte gli elementi presenti nel borsellino
     */
    public void reverse() {
        int coinsSize = coins.size();
        String[] oldCoins = new String[coinsSize];
        oldCoins = coins.toArray(oldCoins);
        for(int i = 0; i < coinsSize; i++) {
            coins.set(i, oldCoins[coinsSize - i - 1]);
        }
    }

    /**
     * Metodo che restituisce il numero di valori presenti nel borsellino
     * @return Numero di valori presenti nel borsellino
     */
    public int getSize() {
        return coins.size();
    }

    /**
     * Metodo che restituisce il valore di una moneta alla posizione 'i'
     * @param i Posizione del valore richiesto
     * @return Valore (String) alla posizione 'i'
     */
    public String getCoinValue(int i) {
        if(i >= coins.size() || i < 0) throw new IllegalArgumentException("La moneta
        all'index" + i + "non è presente nel borsellino");
        return coins.get(i);
    }

    /**
     * Metodo che imposta la moneta ad una posizione 'i' ad un valore 'value'
     * @param i Posizione in cui inserire il valore
     * @param value Valore (String) da inserire nel borsellino
     */
```

```java
    public void setCoinValue(int i, String value) {
        if(i >= coins.size() || i < 0) throw new IllegalArgumentException("La moneta all'index" + i + "non è presente nel borsellino");
        coins.set(i, value);
    }

    /**
     * Metodo che restituisce la posizione della moneta
     * @param value Nome della moneta
     * @return Posizione della moneta
     */
    public int getCoinPosition(String value) {
        return coins.indexOf(value);
    }

    /**
     * Metodo che svuota il borsellino
     */
    public void clear() {
        coins.clear();
    }

    /**
     * Metodo che sposta le monete contenute in un altro borsellino
     * nel borsellino attuale
     * @param other Altro borsellino che viene svuotato
     */
    public void transfer(Purse other) {
        int actualSize = coins.size();
        for(int i = 0; i < other.getSize(); i++) {
            coins.add(actualSize + i, other.getCoinValue(i));
        }
        other.clear();
    }

    /**
     * Metodo che confronta (per contenuto e posizione del contenuto) due borsellini
     * @param other Altro borsellino con cui viene confrontato l'attuale
     * @return True se i due borsellini contengono le stesse monete nella stessa posizione, false se non soddisfano queste condizioni
     */
    public boolean sameContents(Purse other) {
        if(coins.size() == other.getSize()) {
            for(int i = 0; i < coins.size(); i++) {
                if(!coins.get(i).equals(other.getCoinValue(i))) return false;
            }
            return true;
        } else {
            return false;
        }
    }

    /**
     * Metodo che confronta (per contenuto) due borsellini
     * @param other Altro borsellino con cui viene confrontato l'attuale
     * @return True se i due borsellini contengono le stesse monete, false se non soddisfa questa condizione
     */
    public boolean sameCoins(Purse other) {
        int actualSize = coins.size();
        if(actualSize == other.getSize()) {
```

```
119             ArrayList<String> copy = new ArrayList<String>(actualSize);
120             for(int i = 0; i < actualSize; i++) {
121                 copy.add(coins.get(i));
122             }
123             for(int i = 0; i < actualSize; i++) {
124                 if(other.getCoinPosition(coins.get(i)) == -1) {
125                     return false;
126                 } else {
127                     copy.remove(other.getCoinValue(i));
128                 }
129             }
130             if(copy.isEmpty()) return true;
131         }
132         return false;
133     }
134 
135     /**
136      * Metodo toString che restituisce i valori delle monente nel borsellino
137      * @return Testo che indica le monete contenute nel borsellino
138      */
139     @Override
140     public String toString() {
141         return "Purse" + coins.toString();
142     }
143 }
```