Lista klonów list

Stwórz nowy rodzaj kontenera, `ArrayListSqrCloned<E>` która działać będzie podobnie do listy list elmentów typu `E`. Klasa powinna działać podobnie jak `ArrayList<S>` (gdzie `S` jest kolejną listą typu `ArrayList<E>`) zawierając wszystkie jej metody. Można korzystać z `ArrayList` w implementacji naszego kontenera.

Lista powinna różnić się od zwykłej listy list w następujący sposób:

- Wszystkie listy które są wstawiane do kontenera ulegają płytkiemu klonowaniu. Dotyczy to operacji `add`, `addAll` i `set`. Późniejsze zmiany w oryginalnej liście nie są widoczne w liście przechowywanej przez nasz kontener, i vice versa. Niemniej, jeśli poszczególne elementy `E` ulegną zmianie, to takie zmiany są odzwierciedlane wszedzie.

  Operacje `indexOf(arg)` powinny odnajdować kopię, nawet jeśli `arg` jest oryginalną listą (a nie klonem). Podobnie metoda `equals` powinna zwracać `true` dla oryginału i klona, tak długo jeśli ich zawartości rzeczywiście są jednakowe.

- Wszystkie operacje wstawiania do kontenera powinny akceptować nie tylko listy elementów `E` ale też listy po elementach dziedziczonych po `E`. Z drugiej strony, próba wstawienia listy która nie ma z `E` nic wspólnego powinna zakończyć się błędem kompilacji.

  Listy zwracane przez kontener zawsze operują na `E`, nawet jeśli oryginały były po elementach dziedziczonych.

Poniższe metody powinny być zawarte w `ArrayListSqrCloned<E>`. Jeśli nie jest napisane inaczej, ich działanie powinno być takie samo jak w `ArrayList`.

- Bezargumentowy konstruktor tworzący listę pustą
- Konstruktor kopiujący z innego kontenera – podobny do tego z `ArrayList` ale ze ze zmianami wyjaśnionymi powyżej
- `add`, `addAll` i `set` (ze zmianami jak wyjaśnione powyżej)
- `clear`
- `contains` i `containsAll` (z uwagą wyjaśnioną powyżej – zwraca `true` jeśli zawarty jest równoważny klon argumentu)
- `get`
- `indexOf` i `lastIndexOf` (z uwagą wyjaśnioną powyżej)
- `isEmpty`
- `size`
- `iterator` i `listIterator`
- `remove` i `removeAll` (z możliwością usuwania równoważnych klonów)
- `equals` i `hashCode` (z uwagą o klonach wyjaśnioną powyżej)

Pozostałe metody zdefiniowane dla `ArrayList` nie muszą być dostępne w naszym kontenerze.

Przykład

Dla następującego użycia klasy `ArrayListSqrCloned`

```
import java.util.ArrayList;

public class TestListSqr {

  static public class Parent {
      public int x;
  }
  static public class Derived extends Parent {
      public double y;
  }

  static public void main(String[] args) {
      ArrayListSqrCloned<Parent> listsqr = new ArrayListSqrCloned<>();
      ArrayListSqrCloned<Derived> listsqr2 = new ArrayListSqrCloned<>();
      ArrayList<Derived> list = new ArrayList<>();
      list.add(new Derived());
      listsqr.add(list);
      listsqr2.add(list);
      System.out.println(listsqr.equals(listsqr2));
      System.out.println(listsqr.size());
      System.out.println(listsqr2.size());
      listsqr.remove(list);
      System.out.println(listsqr.equals(listsqr2));
      listsqr2.remove(list);
      System.out.println(listsqr.equals(listsqr2));
  }
}
```

Poprawnym wynikiem jest:

```
true
1
1
false
true
```

Wysyłanie na Satori

Wasze rozwiązanie powinno składać się z pojedyńczego pliku `ArrayListSqrCloned.java`. Niektóre z testów celowo mają zakończyć się błędem kompilacji. W takim przypadku:

- Jeśli dla Waszego submitu błąd kompilacji zostanie wykryty, to resultat będzie OK.
- Jeśli błąd kompilacji nie nastąpi, to zgłoszony zostanie specjalny status: “exp:CME got:OK”