Tecnologie dei Linguaggi di Programmazione

Corso di Laurea in Informatica

Soluzioni scritto del 5 Luglio 2010

Esercizio 1

Data la classe seguente:

```
class Point {
   int x;
   int y;
   Point(int x, int y) {
         this.x = x;
         this.y = y;
   }
   public static Point[] makeArray(int n, Point pt) {
         Point[] res = new Point[n];
         for (int i = 0; i < n; i++) {
               pt.x = pt.x + n;
               pt.y = pt.y + n;
               res[i] = pt;
         }
         return res;
   }
   public static Point sum(Point[] a) {
         Point res = a[0];
         for(int i = 1; i < a.length; i++) {
               res.x = res.x + a[i].x;
               res.x = res.x + a[i].y;
         }
         return res;
   }
}
```

Dopo l'esecuzione di:

```
Point pt1 = new Point(1,2);
Point pt2 = Point.sum(Point.makeArray(3, pt1));
```

1.1 Quanto vale pt1.x? **73**
1.2 Quanto vale pt1.y? **11**
1.3 Quanto vale pt2.x? **73**
1.4 Quanto vale pt2.y? **11**

Esercizio 2

Date le classi seguenti:

```
class A {
   String method(B a, B b) {
   return "a";
   }
}

class B extends A {
   String method(A a, B b) {
      return "b";
   }

   String method(B a, A b) {
      return "c";
   }
}
```

e gli oggetti seguenti:

```
A a = new A();
B b = new B();
A c = new B();
```

2.1 Quanto vale a.method(a, a)?
(a) "a"
(b) "b"
(c) "c"
(d) Errore NoSuchMethod
(e) Errore Ambiguous

2.2 Quanto vale b.method(b, b)?
(a) "a"
(b) "b"
(c) "c"
(d) Errore NoSuchMethod
(e) Errore Ambiguous

2.3 Quanto vale c.method(c, c)?
(a) "a"
(b) "b"
(c) "c"
(d) Errore NoSuchMethod
(e) Errore Ambiguous

2.4 Quanto vale b.method(b, c)?
(a) "a"
(b) "b"
(c) "c"
(d) Errore NoSuchMethod
(e) Errore Ambiguous

Esercizio 3

Data la rappresentazione degli alberi (Tree) le cui foglie sono etichettate con una stringa:

```
abstract class Tree {
}

class Node extends Tree{
      Tree[] sons;
      Node(Tree[] sons) {
            this.sons=sons;
      }
}

class Leaf extends Tree {
      String value;
      Leaf(String value) {
            this.value=value;
      }
}
```

aggiungere in `Tree`:

- un metodo `in` che verifica se una stringa è nell'albero;
- un metodo `maxLength` che restituisce una delle stringhe che ha la massima lunghezza o la stringa vuota se l'albero è vuoto;
- un metodo `listOf` che restituisce l'array contenente tutte le stringhe dell'albero.

```
class Integer {
      int i;
      Integer(int i) {
            this.i=i;
      }
      void incr() {
            i++;
      }
      int getI() {
            return i;
      }
}

abstract class Tree {
      abstract boolean in(String s);
      abstract String maxLength();
      abstract int getSize();
      String[] listOf() {
            String[] res = new String[getSize()];
            listOf(res,new Integer(0));
            return res;
      }
      abstract void listOf(String[] res, Integer i);
}
```

```
class Node extends Tree{
	Tree[] sons;
	Node(Tree[] sons) {
		this.sons=sons;
	}
	boolean in(String s) {
		for(int i=0; i<sons.length; i++) {
			if (sons[i].in(s)) {
				return true;
			}
		}
		return false;
	}
	String maxLength() {
		String res = "";
		String s;
		for(int i=0; i<sons.length; i++) {
			s=sons[i].maxLength();
			if (s.length()>res.length()) {
				res=s;
			}
		}
		return res;
	}
	int getSize() {
		int res = 0;
		for(int i=0; i<sons.length; i++) {
			res+=sons[i].getSize();
		}
		return res;
	}
	void listOf(String[] res, Integer i) {
		for(int j=0; j<sons.length; j++) {
			sons[j].listOf(res,i);
		}
	}
}
class Leaf extends Tree {

	String value;
	Leaf(String value) {
		this.value=value;
	}
	boolean in(String s) {
		return value.equals(s);
	}
	String maxLength() {
		return value;
	}
	int getSize() {
		return 1;
	}
	void listOf(String[] res, Integer i) {
		res[i.getI()]=value;
		i.incr();
	}
}
```

Esercizio 4

Scrivere un metodo `checkSym` che prende un array bidimensionale *non necessariamente quadrato* di numeri interi e restituisce un booleano. Tale booleano vale `true` quando c'è una simmetria fra righe e colonne: la riga *i* e la colonna *i* hanno esattamente gli stessi numeri nello stesso ordine per ogni *i*. Per esempio, dato l'array {{1,2,3},{2,1},{3}}, il metodo restituisce true mentre, dato l'array {{1,2,3}}, il metodo restituisce false.

```
boolean checkSym(int [][] a) {
     try {
          for(int i=0; i< a.length; i++) {
               for (int j=0; j<a[i].length; j++) {
                    if (a[i][j]!=a[j][i]) {
                         return false;
                         }
                    }
               }
          }
     catch (IndexOutOfBoundsException e) {
          return false;
     }
     return true;
}
```