Corso di Laurea in Matematica
Corsi di Informatica (n.o.) e Informatica 2 (v.o.)

UNIVERSITÀ DELLA CALABRIA
Dipartimento di Matematica e Informatica

Prova d'Esame del 13/09/2017

Esercizio 1. Si implementi in Java un metodo che legga da input una sequenza di caratteri terminata dal carattere "$". Tale metodo dovrà restituire *true* se la sequenza di caratteri letti è "ben formata" rispetto alle parentesi tonde, *false* altrimenti. Per "ben formata" rispetto alle parentesi tonde si intende qui una condizione semplificata: non può essere aperta una nuova parentesi tonda se prima non è stata chiusa quella aperta in precedenza. In altre parole, ogni parentesi tonda aperta deve essere chiusa, prima di aprire una nuova parentesi e comunque prima di terminare la sequenza.

Esempi:
Sequenze di caratteri "ben formate" per cui il metodo dovrà restituire *true*:
(s d f) d t (g) t y 6 ( ) k (r t y) $
f v 4 g (h j k) $
g h j k y $
f a j ( ) $
Sequenze di caratteri NON "ben formate" per cui il metodo dovrà restituire *false*:
y ((g)) h (b n) $
(h) g j (u f e r $
j g) k s ( ) $
) b) h (k (g $

SOLUZIONE:

```
import java.util.Scanner;
public class VerificaParentesi {
	public static Scanner input = new Scanner(System.in);
	public static boolean verifica() {
		boolean benFormata = true;
		boolean parentesiAperta = false;
		System.out.println("Inserisci una sequenza di caratteri terminata dal carattere '$': ");
		char c = input.next().charAt(0);
		while (c != '$'){
			if (c == '(')
				if (parentesiAperta)
					benFormata = false;
				else
					parentesiAperta = true;
			if (c == ')')
				if (!parentesiAperta)
					benFormata = false;
				else
					parentesiAperta = false;
			c = input.next().charAt(0);
		}
		return benFormata && !parentesiAperta;
	}

	public static void main(String[] args) {
		if (verifica())
			System.out.println("La sequenza è ben formata");
		else
			System.out.println("La sequenza NON è ben formata");
	}
}
```

Esercizio 2. Siano dati due array di numeri interi, "a" e "b", che si suppongono avere sempre la stessa dimensione. Si implementi in Java un metodo RICORSIVO che, ricevuti "a", "b" e un terzo array "c", riempia "c" inserendo in ciascuna posizione il valore massimo tra i due presenti nelle corrispondenti posizioni di "a" e di "b".

Prova d'Esame del 13/09/2017

Esempio: a = {1, 5, 6, 7, 2, 9}
b = {4, 3, 5, 8, 3, 1}
c = {4, 5, 6, 8, 3, 9}

SOLUZIONE:

```
import java.util.*;

public class Inf_20170913 {

	public static Scanner input = new Scanner(System.in);

	public static void creaNuovoArray(int[] a, int[] b, int[] c, int i) {

		System.out.println(a.length + " " + i);
		if (i < a.length) {
			if (a[i] >= b[i])
				c[i] = a[i];
			else
				c[i] = b[i];

			creaNuovoArray(a, b, c, i + 1);
		}
	}

	public static void main(String[] args) {

		// gli array potrebbero anche leggersi da input, non importa la loro provenienza
		int a[] = { 1, 5, 6, 7, 2, 9 };
		int b[] = { 4, 3, 5, 8, 3, 1 };
		int c[] = new int[a.length];

		creaNuovoArray(a, b, c, 0);

		for (int i = 0; i < a.length; i++)
			System.out.print(a[i] + " ");
		System.out.println();
		for (int i = 0; i < a.length; i++)
			System.out.print(b[i] + " ");
		System.out.println();
		for (int i = 0; i < a.length; i++)
			System.out.print(c[i] + " ");
		System.out.println();

	}
}
```

Esercizio 3. Si progetti ed implementi in Java una classe "AlQuadrato" che svolga la funzione di calcolare il quadrato di un numero reale. L'unico dato "custodito" all'interno di ciascuna istanza della classe è la base, che è, appunto, un numero reale. Inoltre, a parte prevedere i metodi "usuali", la classe dovrà essere strutturata e fornire metodi adeguati per le seguenti operazioni:

- *public double calcola()* ← restituisce un numero reale pari al quadrato della base memorizzata nell'istanza
- *public double getBase()* ← restituisce la base memorizzata nell'istanza

Prova d'Esame del 13/09/2017

- *public void setBase(double x)* ← modifica la base memorizzata nell'istanza
- *public String toString()* ← esempio di stringa restituita, se la base fosse "3.1": 3.1^2=9.61
- *public boolean equals(Object obj)* ← metodo standard
- *public boolean maggioreDi (AlQuadrato a)* ← restituisce "true" se l'istanza corrente ha una base maggiore di quella dell'oggetto "a".

SOLUZIONE:

```
public class AlQuadrato {
	private double base;
		public AlQuadrato() {
		base = 1.0;
	}
		public AlQuadrato(double x) {
		base = x;
	}
		public double calcola() {
		return base * base;
	}
		public double getBase() {
		return base;
	}
		public void setBase(double x) {
		this.base = x;
	}
		public String toString() {
		String s = "";
		s += this.base;
		s += "^2 = ";
		s += this.calcola();
		return s;
	}

	@Override
	public boolean equals(Object obj) {
		if(!(obj instanceof AlQuadrato))
			return false;

		AlQuadrato c = (AlQuadrato) obj;
		return this.base == c.base;
	}

	public boolean maggioreDi (AlQuadrato a) {
		return this.base>a.base;
	}
}
```