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 02/02/2017

Esercizio 1. Si implementi in Java una classe per la gestione (semplificata) della collezione di farfalle di un appassionato di entomologia. La classe sarà denominata "GestioneCollezioneFarfalle": si assuma che ogni istanza della classe dovrà riuscire a gestire *al massimo 5000 farfalle*. In particolare, a parte prevedere i metodi "usuali", la classe dovrà essere strutturata e fornire metodi adeguati per le seguenti operazioni:

- *public boolean inserisci (Farfalla f)* ← Registrare (inserire) una nuova farfalla - restituisce "false" se la struttura che contiene le farfalle è già piena (quindi sono già presenti cinquemila farfalle).
- *public void stampa()* ← Stampare su standard output (non restituire!) tutte le farfalle memorizzate.
- public int nFarfalle() ← Restituire il numero di farfalle comprese nella collezione.
- *public Farfalla[] farfalleDiColore (int anno)* ← Dato un anno, restituire un array (non stampare su output!) con tutte le farfalle catturate in quell'anno e attualmente comprese nella collezione.
- *public Farfalla farfallaPiuGrande(Sring colore)* ← Dato un colore, restituire (non stampare su output!) la farfalla più grande tra quelle di quel colore comprese nella collezione (si può supporre che ci sia sempre una unica farfalla di dimensione massima).

Ogni farfalla è caratterizzata dai seguenti dati: specie (per semplicità, rappresentata da una stringa), colore predominante (per semplicità, rappresentata da una stringa), anno in cui è stata catturata (per semplicità, rappresentato da un numero intero), grandezza (per semplicità, rappresentata da un numero intero compreso tra 1 e 100). Si suggerisce l'implementazione di una classe apposita, che potrebbe essere denominata "Farfalla", per modellare le farfalle come descritto; naturalmente, si è liberi di implementare eventuali altre classi o metodi ritenuti utili.

SOLUZIONE:

```
public class Farfalla {
	private String specie;
	private String colore;
	private int anno;
	private int grandezza;

	public Farfalla(String specie, String colore, int anno, int grandezza) {
		this.specie = specie;
		this.colore = colore;
		this.anno = anno;
		this.grandezza = grandezza;
	}

	public String toString() {
		return "Farfalla: specie:" + specie + " colore:" + colore + " anno:" + anno
+ " grandezza" + grandezza;
	}

	public String geSpecie() {
		return specie;
	}

	public String getColore() {
		return colore;
	}

	public int getAnno() {
		return anno;
	}

	public int getGrandezza() {
		return grandezza;
```

Prova d'Esame del 02/02/2017

```
	}
}

public class GestioneCollezioneFarfalle {

	private Farfalla[] collezione;
	private int conta = 0;

	public GestioneCollezioneFarfalle() {
		this.collezione = new Farfalla[5000];
		this.conta = 0;
	}

	public int getNumeroMacchine() {
		return this.conta;
	}

	public int nFarfalle() {
		return conta + 1;
	}

	public boolean inserisci(Farfalla a) {

		if (conta > 4999)
			return false;
		else {
			collezione[conta] = a;
			conta++;
			return true;
		}
	}
	public void stampa() {
		for (int i = 0; i < conta; i++) {
			System.out.println(collezione[i]);
		}
	}
	public Farfalla[] farfalleDiColore(int anno) {
		// nota: restituisce un array potenzialmente mezzo vuoto.
		Farfalla[] res = new Farfalla[5000];
		int k = 0;
		for (int i = 0; i < conta; i++) {
			if (collezione[i].getAnno() == anno)
				res[k] = collezione[i];
			k++;
		}
		return res;
	}

	public Farfalla farfallaPiuGrande(String colore) {
		Farfalla max = null;

		for (int i = 0; i < conta; i++) {
			if (collezione[i].getColore().equals(colore)) {
				if (max == null || collezione[i].getGrandezza() >
max.getGrandezza())
					max = collezione[i];
			}
```

Prova d'Esame del 02/02/2017

```
            }
            return max;
      }
}
```

Esercizio 2. Si implementi in Java un metodo che riceva come parametri due matrici di caratteri (siano **M1** ed **M2**) ed un array di numeri interi (sia **a**) e restituisca un numero reale, comportandosi come descritto di seguito. Si può supporre che nella matrice M1 siano presenti caratteri unici, senza ripetizioni; per ciascun carattere **c** presente in M1 e presente *anche in M2*, si contino le occorrenze di c in M2 (cioè quante volte c compare in M2), e sia questo valore **n**; si calcoli un valore **f** per il carattere c, in accordo ai due casi possibili:

- **n** è presente nell'array **a**: in questo caso, f=n;
- **n** non è presente nell'array **a**: in questo caso, f=0 (zero).

Il metodo restituirà la media dei valori f calcolati per tutti i caratteri presenti in M2.

SOLUZIONE:

```
public float ex2(char[][] m1, char[][] m2, int[] a) {

            int rm1 = m1.length;// righe
            int cm1 = m1[0].length;// colonne
            float sommaF = 0;// risultato finale
            int charPresenti = 0;
            for (int i = 0; i < rm1; i++) {
                  for (int j = 0; j < cm1; j++) {
                        char c = m1[i][j];
                        // conta occorrenze di c in m2
                        int n = contaOccorrenze(m2, c);
                        if (n > 0) {// se c esiste in m2
                              if (contiene(a, n)) {// se a contiene n allora f=n
                                    sommaF += n;
                              }
                              charPresenti++;
                        }
                  }
            }
            if (charPresenti == 0)
                  return sommaF;// evita divisione per zero
            return sommaF / charPresenti;// media dei valori f per tutti i caratteri
presenti in m2.
      }

      boolean contiene(int[] a, int n) {
            for (int j = 0; j < a.length; j++) {
                  if (a[j] == n)
                        return true;
            }
            return false;
      }
      int contaOccorrenze(char[][] m2, char daTrovare) {
            int n = 0;
            for (int i = 0; i < m2.length; i++) {
                  for (int j = 0; j < m2[0].length; j++) {
                        if (m2[i][j] == daTrovare)
                              n++;
```

Prova d'Esame del 02/02/2017

```
                }
            }
            return n;
        }
```