Corso di laurea in Comunicazione digitale
AA 2009/10

Programmazione I
Prof. M. A. Alberti

II Compitino - 18 dicembre 2009

---

**Cognome** **Nome**

**Matricola**

---

**1. (2 punti)**
Definite una classe **`Parole`** con un campo **`testo`**, di tipo array di stringhe di caratteri (istanze della classe **`String`**) e un campo di tipo intero **`lung`** che indica la lunghezza effettiva dell'array. La classe deve fornire anche una costante di tipo intero **`MAX`** per indicare la dimensione massima dell'array.

```
public class Parole {
   final int MAX = 5;
   private String[] testo;
   private int lung;
   …
}
```

---

**2. (1 punto)**
Istanziare il campo **`testo`** usando la costante **`MAX`**

```
testo = new String[MAX];
```

---

**3. (5 punti)**
Definire un costruttore di oggetti della classe **`Parole`** che mediante un ciclo **`while`** acquisisca le parole per inizializzare il campo **`testo`**. Si controlli ovviamente di non superare la dimensione massima dell'array e si consideri che il tasto **<return>** segnala l'intenzione di non fornire più parole. Il costruttore, acquisite le parole da input inizializza anche il campo **`lung`** in modo adeguato.

```
Parole() {
   ConsoleInputManager in = new ConsoleInputManager();

   String vuota = "";
   String parola;
   int cont = 0;

   while (!(parola=in.readLine("parola? ")).equals(vuota) &&
                                                  cont < MAX){
               testo[cont++] = parola;
   }

   lung = cont;
}
```

**4. (3 punti)**

Definire il metodo d'istanza **`stampaTesto()`** per produrre in output l'elenco su diverse righe delle parole archiviate e della loro lunghezza

```
public void stampaTesto() {
   ConsoleOutputManager out = new ConsoleOutputManager();
   out.println("stampo… testo e lunghezza parole…");

   for (int i=0; i<lung; i++)
      out.println(testo[i] + "\t" + testo[i].length());
}
```

**5. (6 punti)**

Definire il metodo d'istanza **`espandiTesto()`** (prototipo e corpo) che riporta un array di dimensione adeguata inizializzato con le parole archiviate nel campo **`testo`**, intervallate dalle stesse parole in caratteri maiuscoli. Ad es.: nel caso **`MAX=10`** il campo potrebbe essere rappresentato dall'array:

**`[Sempre, caro, mi, fu, quest'ermo, colle, null, null, null, null]`**

il nuovo array creato dal metodo **`espandiTesto()`** sarà

**`[Sempre, SEMPRE, caro, CARO, mi, MI, fu, FU, quest'ermo, QUEST'ERMO, colle, COLLE]`**

```
public String[] espandi Testo() {
   String[] nuovo Testo = new String[2 * lung];

   for (int i=0; i<lung; i++) {
      nuovo Testo[2 * i] = testo[i];
      nuovo Testo[2 * i + 1] = testo[i].toUpperCase();
   }

   return nuovo Testo;
}
```

**6. (6 punti)**

Nella classe **`Parole`** sovraccaricate il costruttore con **`Parole(String riga)`** che riceve in input una riga con parole separate da spazi bianchi e inizializza, come nel caso precedente, il campo **`testo`** usando un'istanza della classe **`StringTokenizer`** per processare le singole parole.

```
Parole(String riga) {
   StringTokenizer tokenizer = new StringTokenizer(riga);
   int cont = 0;

   while(tokenizer.hasMoreTokens() && cont < MAX)
      testo[cont++] = tokenizer.nextToken();

   lung = cont;
}
```

**7. (3 punti)**

Supponete di avere le variabili booleane: **`gioco studio pioggia lavoro`**
e l'istruzione condizionale:

```
if (!pioggia&&gioco) if (studio||lavoro)
out.println("pazienza"); else out.println("il momento giusto");
else out.println("un po' di fatica");
```

Formattate l'istruzione in modo da renderla più leggibile:

```
if (!pioggia&&gioco)
    if (studio||lavoro)
      out.println("pazienza");
    else out.println("il momento giusto");
else out.println("un po' di fatica");
```

e compilate la tabella di verità seguente in tutti i suoi casi, usando I simboli T e F:

| | gioco | studio | pioggia | lavoro | !pioggia | !pioggia&&gioco | studio\|\|lavoro |
|---|---|---|---|---|---|---|---|
| **0** | F | F | F | F | T | F | F |
| 1 | F | F | F | T | T | F | T |
| 2 | F | F | T | F | F | F | F |
| 3 | F | F | T | T | F | F | T |
| 4 | F | T | F | F | T | F | T |
| 5 | F | T | F | T | T | F | T |
| 6 | F | T | T | F | F | F | T |
| 7 | F | T | T | T | F | F | T |
| 8 | **T** | **F** | **F** | **F** | **T** | **T** | **F** |
| 9 | T | F | F | T | T | T | T |
| 10 | T | F | T | F | F | F | F |
| 11 | T | F | T | T | F | F | T |
| 12 | T | T | F | F | T | T | T |
| 13 | T | T | F | T | T | T | T |
| 14 | T | T | T | F | F | F | T |
| 15 | T | T | T | T | F | F | T |

Per poter affermare per quali valori delle variabili verrà stampata "*il momento giusto*"

| gioco | studio | pioggia | lavoro |
|---|---|---|---|
| T | F | F | F |

**8. (4 punti)**

Date le classi seguenti: A classe base e B sua sottoclasse B:

```
class A{
   protected int a;
   private int b;
   A(int n){
      a = n;
      b = 1;
   }
   public int set_b(int n){
      return b = n;
   }
   public int get_b(){
      return b;
   }
}
```

```
class B extends A{
   public int b;
   B(int n){

      super(n);
      b = n;

   }
   public int set_b(int n){
      return b = 2 * n;
   }
}
```

a. completare nel riquadro il codice del costruttore **`B(int)`**
b. dire se la doppia dichiarazione del campo **`b`** nella classe base e nella sottoclasse sono in conflitto SI **NO**
c. definita la variabile **`B y = new B(num)`** dire se è lecita e perchè l'istruzione **`y.get_b()`**
   l'istruzione è lecita perchè il metodo **`get_b()`** è ereditato dalla classe **`A`**. Essendo però un metodo di classe **`A`**, legge il campo **`b`** della classe anche quando è chiamato su oggetti della classe **`B`**. Ricordatevi che gli oggetti di classe **`B`** incorporano nella propria struttura anche quella della classe **`A`**, inclusi i campi privati non direttamente accessibili.

d. dire se costituisce errore o altro la definizione di **`int set_b(int)`** nella classe **`B`**
   la definizione del metodo con prototipo **`int set_b(int)`** nella classe **`B`** non costituisce errore ma una sovrascrittura dello stesso metodo nella classe **`A`**

---

**9. (4 punti)**

Il seguente spezzone di codice fa parte di un metodo della classe **`A`** dell'esercizio precedente:

```
A x = new A(15);
x.set_b(x.a*2 + x.get_b());
```

Valutate l'espressione:

`x.get_b()` **31**

```
B y = new B(14);
```

Valutate l'espressione:

`y.get_b()` **1**

```
y.set_b(24);
```

Valutate le espressioni:

`y.get_b()` **1**

`y.b` **48**