## Commenti sull'esercizio 1

Era richiesto di realizzare **due metodi ricorsivi**, leggere da standard input una stringa e riportare in uscita la concatenazione delle sole vocali in ordine in cui appaiono (output del primo metodo) e in ordine inverso (output del secondo metodo).

La lettura da standard input non ha creato grossi problemi, quindi mi focalizzo sui due metodi ricorsivi, partendo dal primo.

Innanzitutto **era richiesto che il metodo restituisse la concatenazione delle vocali**, quindi chi ha stampato le vocali senza costruire la stringa da restituire non ha risposto ai requisiti.

Quando si maneggiano degli oggetti è sempre bene **verificare come prima cosa che il riferimento non sia null**, altrimenti qualsiasi chiamata ai loro metodi lancerà NullPointerException. Pochi si sono ricordati questo controllo.

Il problema va suddiviso in **caso base e chiamata ricorsiva**. Nel caso specifico il caso base corrisponde alla stringa vuota, che non contiene vocali e pertanto restituirà come valore in output proprio la stringa vuota. Chi non ha incluso il caso base ha commesso un errore grave.

Se ho una stringa con almeno un carattere dovrò scomporla in modo da effettuare delle chiamate ricorsive su problemi di taglia più piccola. La soluzione più semplice era analizzare il primo carattere e, se era una vocale si restituiva la concatenazione tra quel carattere e la stringa restituita dalla chiamata ricorsiva sulla parte restante della stringa. Altrimenti ci si limitava a restituire la stringa risultante dalla chiamata ricorsiva sul resto della stringa.

Per l'analisi del carattere **io ho usato una variabile di tipo char** attribuendole il valore restituito dal metodo **chatAt(index)** della classe String. In questo caso i confronti vanno fatti con "**==**".

```
if  (c=='a' || c=='e' || c=='i' || c=='o' || c=='u' ||
     c=='A' || c=='E' || c=='I' || c=='O' || c=='U'){
     return c+stampaVocali(s.substring(1));
}
else{
     return stampaVocali(s.substring(1));
}
```

Molti hanno **estratto una sottostringa di lunghezza 1** con String c = s.substring(0,1) e poi utilizzato un'espressione booleana di or con confronti tipo **c.equals("a")** per ogni vocale maiuscola e minuscola, oppure **c.equalsIgnoreCase("a")** per le sole minuscole (o maiuscole), riducendo il numero di confronti. Andava benissimo. Chi ha estratto una sottostringa e confrontato con "==" ha commesso un errore concettuale grave perché' "==" confronta i riferimenti, non il contenuto delle stringhe.

Qualcuno non ha utilizzato un'espressione booleana ma ha utilizzato una successione di i**f annidati**. Questa soluzione, anche se funzionante, non è ottima anche perché' l'istruzione da invocare, in caso di vocale è sempre la stessa:

```
if(c==‘a’)
      return c+stampaVocali(s.substring(1)); //uguale
else if (c==‘e’)
      return c+stampaVocali(s.substring(1)); //uguale
else if (c==‘i’)
      return c+stampaVocali(s.substring(1)); //uguale
etc.
```

Meglio chi ha usato uno switch, che almeno limitava le ripetizioni sfruttando il fatto che se una condizione è verificata si eseguono tutte quelle sottostanti se non si mette un break. La struttura pertanto era meno ripetitiva:

```
switch(c) {
      case 'a':
      case 'A':
      case 'e':
      case 'E':
      case 'i':
      case 'I':
      case 'o':
      case 'O':
      case 'u':
      case 'U': return c+stampaVocali(s.substring(1));
      default:  return stampaVvocali(s.substring(1));
}
```

Per quanto riguarda il secondo metodo il ragionamento era del tutto simile solo bisognava stare attenti al carattere che si estraeva e a come lo si concatenava. In particolare, se si analizza il primo carattere, se questo è una vocale, si restituisce la stringa risultato della chiamata ricorsiva sulla sottostringa dalla seconda posizione in poi, concatenata al carattere stesso. Se si analizza l'ultimo carattere e questo è una vocale, si restituisce la concatenazione del carattere seguito dal risultato della chiamata ricorsiva sulla sottostringa che parte dall'inizio e arriva al penultimo carattere.

```
c = charAt(0);
if  (c=='a' || c=='e' || c=='i' || c=='o' || c=='u' ||
     c=='A' || c=='E' || c=='I' || c=='O' || c=='U'){
      return stampaVocaliRev(s.substring(1))+c;
}
else{
      return stampaVocaliRev(s.substring(1));
}
```

Oppure:

```
c = charAt(s.length()-1);
if  (c=='a' || c=='e' || c=='i' || c=='o' || c=='u' ||
     c=='A' || c=='E' || c=='I' || c=='O' || c=='U'){
      return c+stampaVocaliRev(s.substring(0,s.length()-1));
}
else{
      return stampaVocaliRev(s.substring(0,s.length()-1));
}
```

Entrambe le soluzioni andavano bene. Gli **errori più comuni** sono stati fare un **mix dei due possibili approcci** (es. estrarre l'ultimo carattere ma concatenarlo alla fine, oppure invocare ricorsivamente sulla sottostringa sbagliata) **e, soprattutto, nella chiamata ricorsiva invocare il primo metodo!** Questo errore probabilmente è dovuto a cut&paste del primo metodo in cui vi siete dimenticati di aggiornare anche il nome del metodo! E' un errore subdolo, ma in output vi viene il risultato sbagliato, quindi avete la possibilità di correggere!