Programmazione per Bioinformatica Anno Accademico 2013/2014 **Prova di Esame del 20 Giugno 2014**	*tempo a disposizione: 3 ore* *consegnare solo la 'bella'*

– Stringhe –

Esercizio 1 [5 punti]

Si scriva un metodo

```
public static boolean valida(String s)
```

che determina se la stringa `s` è un'alternanza di cifre e non-cifre (una cifra, una non cifra, una cifra, una non cifra, ecc.) ed ha lunghezza pari. Per esempio:
`3e6y2#0i5u` soddisfa tale proprietà,
`3e6y2#0i5u1` *non* soddisfa tale proprietà,
`e6y2#0i5u3` *non* soddisfa tale proprietà,
`36yx2#0i5u` *non* soddisfa tale proprietà.

Suggerimento Per capire se un carattere rappresenta una cifra, si può usare il seguente metodo della classe `Character`

```
public static boolean isDigit(char ch)
```

che restituisce `true` se e solo se il carattere `ch` è uno tra: `'0','1','2','3','4','5','6','7','8','9'`.

Esercizio 2 [7 punti]

Si scriva un metodo

```
public static String espandi(String s)
```

che controlla se `s` è un'alternanza di cifre e non cifre (invocando il metodo `valida`) e in caso contrario ritorna `null`. Altrimenti ritorna una nuova stringa ottenuta ripetendo le non-cifre il numero di volte indicato dalla cifra alla loro sinistra. Per esempio, se `s` è `3e6y2#0i5u` il risultato sarà la stringa `eeeyyyyyy##uuuuu`

Suggerimento. Per convertire un carattere numerico nell'intero corrispondente si può usare il seguente metodo statico della classe `Character`

```
public static int getNumericValue(char ch)
```

che restituisce il valore intero rappresentato dal carattere `ch`. Per esempio, se invocato con argomento `'3'` il metodo restituisce il valore intero 3.

– Ricorsione –

Esercizio 3 [7 punti]

Si definisca il *metodo ricorsivo* (non è ammesso l'uso di cicli!)

```
boolean vettoreCrescente(int[] vett, int pos)
```

Che riceve in input un array di interi `vett` e un intero `pos` *positivo* (maggiore o uguale a 0). Il metodo restituisce `true` se i valori che vanno dalla posizione `pos` (compresa) fino alla fine di `vett` sono in ordine crescente, `false` altrimenti.

Per esempio, l'esecuzione del seguente `main`

```
public static void main(String[] args) {
      int[] vett1 = { 1, 2, 4, 6, 5, 12, 3, 4, 6, 20 };
      int[] vett2 = { 1, 2, 6, 6, 8, 12, 13, 24, 36, 36 };

      System.out.println("Primo vettore crescente: " + vettoreCrescente(vett1, 0));
      System.out.println("Primo vettore crescente: " + vettoreCrescente(vett1, 6));
      System.out.println("Secondo vettore crescente: " + vettoreCrescente(vett2, 0));
}
```

deve stampare a video

```
Primo vettore crescente: false
Primo vettore crescente: true
Secondo vettore crescente: true
```

– Classi –

Si considerino le seguenti interfacce che rappresentano insiemi di stringhe:

```
public interface Set {
  String[] getElements()
  boolean contains(String element);
  boolean intersects(Set that);
  int size();
}

public interface ModifiableSet extends Set {
  boolean add(String element);
  boolean remove(String element);
  boolean addAll(Set set);
  boolean removeAll(Set set);
}
```

La prima specifica l'interfaccia di un insieme senza metodi di modifica, mentre la seconda quella di un insieme con metodi di modifica. L'idea da seguire per l'implementazione è la seguente:

- Il metodo `getElements` restituisce un array con tutti gli elementi dell'insieme.
- Il metodo `contains` determina se `element` è contenuto nell'insieme.
- Il metodo `intersects` determina se `this` interseca `that`.
- Il metodo `size` restituisce il numero di elementi contenuti nell'insieme.
- Il metodo `add` aggiunge un elemento, il metodo `remove` rimuove un elemento. Essi restituiscono `true` se e solo se viene effettivamente aggiunto o rimosso l'elemento. Per esempio, se si aggiunge un elemento a un insieme che già lo contiene, `add` non aggiunge nulla e deve restituire `false`; se si rimuove un elemento da un insieme che non lo contiene, `remove` non rimuove nulla e deve restituire `false`.
- Il metodo `addAll` aggiunge a `this` tutti gli elementi di `set` che non compaiono in `this` e restituisce `true` se e solo se aggiunge effettivamente almeno un elemento.
- Il metodo `removeAll` toglie da `this` tutti gli elementi di `set` che compaiono in `this` e restituisce `true` se e solo se rimuove effettivamente almeno un elemento.

Esercizio 4 [7 punti] - **Solo per questo esercizio, commentate il codice in stile** `javadoc` **-**

Si scriva una classe `ArraySet` che implementa `Set`. Per memorizzare gli elementi dell'insieme, la classe `ArraySet` utilizza il campo

```
private String[] elements;
```

e definisce anche i seguenti costruttori e metodi:

```
public ArraySet(String... elements)
protected void setElements(String[] elements)
public boolean equals(Object that)
```

In dettaglio:

- Il costruttore costruisce un insieme non modificabile che contiene gli elementi forniti.
- Il metodo `setElements` fornisce accesso in scrittura ad `elements`.
- Il metodo `equals` determina se `this` e `that` sono due `Set` che contengono gli stessi elementi, in qualsiasi ordine.
- L'uguaglianza fra gli elementi degli insiemi deve essere determinata dal metodo `equals` di tali elementi, non dall'operatore ==.

Esercizio 5 [7 punti]

Si scriva una classe `ModifiableArraySet` che implementa `ModifiableSet` ed estende `ArraySet`. La classe definisce inoltre i seguenti costruttori:

```
public ModifiableArraySet()
public ModifiableArraySet(String... elements)
public ModifiableArraySet(Set father)
```

- Il costruttore senza argomenti costruisce un insieme inizialmente vuoto.
- Il costruttore che riceve come argomento un altro insieme `father` costruisce un insieme modificabile che contiene inizialmente gli stessi elementi di `father`.

Se tutto è corretto, l'esecuzione del seguente programma:

```
public class Main {

  public static void main(String[] args) {
    Set s1 = new ArraySet("ciao", "amico", "come", "va?");
    Set s2 = new ModifiableArraySet("oggi", "va?", "bene", "male");

    System.out.println("1: " + s1.equals(s2));
    System.out.println("2: " + s1.intersects(s2));

    ModifiableSet s3 = new ModifiableArraySet("amico", "va?", "ciao", "va?");
    s3.add("come");
    s3.add("ci" + "ao");

    System.out.println("3: " + s1.equals(s3));
    System.out.println("4: " + s1.intersects(s3));
  }
}
```

dovrà stampare:

```
1: false
2: true
3: true
4: true
```

Appendice:
Java Platform, Standard Edition 7 API Specification for Class `String`

`char charAt(int index)`: Returns the char value at the specified index.

`int compareTo(String anotherString)`: Compares two strings lexicographically.

`int compareToIgnoreCase(String str)`: Compares two strings lexicographically, ignoring case differences.

`String concat(String str)`: Concatenates the specified string to the end of this string. If the length of the argument string is `0`, then this `String` object is returned. Otherwise, a new `String` object is created, representing a character sequence that is the concatenation of the character sequence represented by this `String` object and the character sequence represented by the argument string.

`boolean equals(Object anObject)`: Compares this string to the specified object.

`boolean equalsIgnoreCase(String anotherString)`: Compares this `String` to another `String`, ignoring case considerations.

`int indexOf(int ch)`: Returns the index within this string of the first occurrence of the specified character.

`int indexOf(int ch, int fromIndex)`: Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index.

`int indexOf(String str)`: Returns the index within this string of the first occurrence of the specified substring.

`int indexOf(String str, int fromIndex)`: Returns the index within this string of the first occurrence of the specified substring, starting at the specified index.

`boolean isEmpty()`: Returns true if, and only if, `length()` is `0`.

`int lastIndexOf(int ch)`: Returns the index within this string of the last occurrence of the specified character.

`int lastIndexOf(int ch, int fromIndex)`: Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index.

`int lastIndexOf(String str)`: Returns the index within this string of the last occurrence of the specified substring.

`int lastIndexOf(String str, int fromIndex)`: Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index.

`int length()`: Returns the length of this string.

`String replace(char oldChar, char newChar)`: Returns a new string resulting from replacing all occurrences of `oldChar` in this string with `newChar`.

`String replace(CharSequence target, CharSequence replacement)`: Replaces each substring of this string that matches the literal `target` sequence with the specified literal `replacement` sequence.

`String replaceAll(String regex, String replacement)`: Replaces each substring of this string that matches the `regex` with the given `replacement`.

`String replaceFirst(String regex, String replacement)`: Replaces the first substring of this string that matches the `regex` with the given `replacement`.

`boolean startsWith(String prefix)`: Tests if this string starts with the specified `prefix`.

`boolean startsWith(String prefix, int toffset)`: Tests if the substring of this string beginning at the specified index starts with the specified `prefix`.

`String substring(int beginIndex)`: Returns a new string that is a substring of this string. The substring begins with the character at the specified index and extends to the end of this string.

`String substring(int beginIndex, int endIndex)`: Returns a new string that is a substring of this string. The substring begins at the specified `beginIndex` and extends to the character at index `endIndex - 1`. Thus the length of the substring is `endIndex - beginIndex`.

`String toLowerCase()`: Returns the `String` object obtained by converting all of the characters in this `String` to lower case.

`String toUpperCase()`: Returns the `String` object obtained by converting all of the characters in this `String` to upper case.

`String trim()`: Returns a copy of this string, with leading and trailing whitespace omitted.