Programmazione per Bioinformatica Anno Accademico 2013/2014 **Prova di Esame del 16 Febbraio 2015**	*tempo a disposizione: 3 ore* *consegnare solo la 'bella'*

– Stringhe –

Esercizio 1 [8 punti]

Si scriva un metodo

```
public static String camelize(String s)
```

che riceve come parametro una stringa `s` e restituisce una nuova stringa, che è la versione *camel-style* della stringa `s`, in cui cioè gli spazi sono stati eliminati e l'inizio delle parole (tranne la prima) è stato scritto in maiuscolo. Si assuma per semplicità che `s` contenga solo caratteri alfabetici minuscoli e spazi, non abbia mai due spazi di seguito e cominci e termini con un carattere alfabetico minuscolo.

Per esempio, l'esecuzione del seguente `main`

```
public static void main(String[] args) {
   System.out.println(camelize("camels are sweet and clever animals"));
}
```

deve stampare a video: `camelsAreSweetAndCleverAnimals`.

– Ricorsione –

Esercizio 2 [8 punti]

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

```
      private static int contaPari(int[] vett, int pos)
```

che riceve in input un array di interi `vett` e un intero `pos`. Il metodo restituisce il numero di valori pari che compaiono in `vett` dalla posizione `pos` (compresa) fino alla fine. Nel caso il parametro `pos` non rappresenti un indice valido, il metodo restituisce `0`.

Per esempio, l'esecuzione del seguente `main`

```
public static void main(String[] args) {
   int[] vett = { 23, 45, 32, 9, 23, 8, 9, 23, 45, 8 };

   System.out.println(
      contaPari(vett, 0) + ", " +
      contaPari(vett, 3) + ", " +
      contaPari(vett, -3));
}
```

deve stampare a video: `3, 2, 0`

– Classi –

Esercizio 3 [8 punti] - **commentate il codice in stile** `javadoc` -

Come ben sapete, una settimana è divisa in 7 *giorni*, secondo la seguente tabella:

nella settimana	nome in italiano	nome in inglese
posizione 0	Domenica	Sunday
posizione 1	Lunedì	Monday
posizione 2	Martedì	Tuesday
posizione 3	Mercoledì	Wednesday
posizione 4	Giovedì	Thursday
posizione 5	Venerdì	Friday
posizione 6	Sabato	Saturday

Si completi la seguente classe che rappresenta un giorno della settimana:

```
public abstract class Giorno {
   // inserire qui eventuali campi

   protected Giorno(int posizione) {
      // inserire qui il corpo del metodo
   }

   public abstract Giorno seguente(int giorniTrascorsi);

   public int getPosizione() {
      // inserire qui il corpo del metodo
   }

   @Override
   public abstract String toString();
}
```

dove `getPosizione()` restituisce la posizione nella settimana, che era stata passata al costruttore. Il metodo `seguente()`, che deve essere implementato nelle sottoclassi concrete, restituisce il giorno che segue `this` dopo tanti giorni quanti indicati dal parametro `giorniTrascorsi`. Si noti che dopo sabato si riparte dalla domenica; per esempio: il quarto giorno dopo un venerdì è il martedì.

Si scrivano poi le sottoclassi concrete `GiornoIT` e `GiornoUK` di `Giorno` che si differenziano per il metodo `toString()`, che restituisce il nome del giorno, rispettivamente, in italiano e in inglese.

Esercizio 4 [8 punti]

Si completi la seguente classe. Un `Corso` universitario è rappresentato dalla `materia` e dai `giorni` della settimana in cui c'è lezione.

```
public class Corso {
   // inserire qui eventuali campi

   public Corso(String materia, Giorno... giorni) {
      // inserire qui il corpo del costruttore
   }

   public String getMateria() {
      // inserire il corpo del metodo
   }

   public Corso getCorsoPosticipato(int posticipo) {
      // inserire il corpo del metodo
   }

   @Override
   public final String toString() {
      // inserire il corpo del metodo
   }
}
```

Il metodo `getMateria()` restituisce la materia del corso. Il metodo `toString()` restituisce una stringa con la materia del corso e i giorni delle settimana in cui c'è lezione. Il metodo `getCorsoPosticipato()` restituisce un `Corso` della stessa materia di `this` posticipato di `posticipo` giorni (ovvero: i giorni di lezione sono tutti stati posticipati di `posticipo` posizioni nella settimana rispetto a quelli di `this`).

Se tutto è corretto, il seguente programma:

```
public class Main {
   public static void main(String[] args) {
      Corso progBioInfo = new Corso("Programmazione per Bioinformatica",
            new GiornoIT(1), new GiornoIT(3));
      Corso asd = new Corso("Algorithms and Data Structures", new GiornoUK(2),
            new GiornoUK(3), new GiornoUK(5));

      System.out.println(progBioInfo);
      System.out.println(asd);
      System.out.println(progBioInfo.getCorsoPosticipato(2));
      System.out.println(asd.getCorsoPosticipato(6));
   }
}
```

stamperà:

```
Programmazione per Bioinformatica: Lunedì Mercoledì

Algorithms and Data Structures: Tuesday Wednesday Friday

Programmazione per Bioinformatica: Mercoledì Venerdì

Algorithms and Data Structures: Monday Tuesday Thursday
```

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, or `-1` if the character does not occur.

`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.