Problem 1

Create a class **MyString**, objects of which describe strings of characters, and have only one private field str of type **String**. In the class define:

- default (parameterless) constructor, which sets the value of str to empty string (but *not* **null**);
- constructor taking a **String**; the value of the argument becomes the value of str;
- method `getLength()` returning the length of the current value of str;
- method `getChar(int n)` returning n-th character (zero based) of str. If the argument is illegal, the method throws (unchecked) exception **IllegalArgumentException**;
- method `append(String s)` appending s to str;
- method `append(int rep, String s)` appending rep repetitions of s to str;
- method `prepend(String s)` prepending s to str;
- method `insert(int pos, String s)` inserting s into str at position pos; for example, if current str is `"abcdef"`, then after inserting `"123"` at position 2, one should get `"ab123cdef"`. If the argument is illegal, the method throws (unchecked) exception **IllegalArgumentException**;
- method `reset(String s)` which substitutes s for str;
- redefinition of the **toString** method from class **Object**.

Do not use any classes from packages other than ***java.lang***. Remeber that objects of class **String** are immutable!

Problem 2

Write static functions operating on and returning **String**s:

```
1    public static String norm(String name)
2    public static String init(String name)
3    public static String tr(String s, String from, String to)
```

where

- the first returns a string which is similar to name, but the first letter is always in upper case and all the remaining characters are in lower case (e.g., `"jOhN"` → `"John"`);
- the second takes a full name, i.e., first name, perhaps a middle name and the lastname, separated by exactly one space; it returns a string in which first and middle name are replaced by the initials (upper case, with a dot after the letter), and the last name starts with a capital letter with all the remaining letters in lower case (e.g. `"john richard doe"` → `"J. R. Doe"`). Note: method **split** from class **String** splits a string into parts separated by a separator which is passed as an argument and returns them as an array of **String**s, e.g., after

```
String[] a = "abc def ghi".split(" ");
```

the array a will contain three elements: "abc", "def" and "ghi".

- the third returns a **String** in which all characters from s that are present in from are replaced by the corresponding (on the same position) characters from to. For this to make sense, all characters in from must be different and from and to should be of the same length. For example, if from is "abc" and to "XXY", then all occurrences of 'a' and 'b' should be replaced by 'X' and 'c' by 'Y'.

For example, the program

download *StringMisc.java*

```
public class StringMisc {
    public static String norm(String name) { ... }
    public static String init(String name) { ... }
    public static String tr(String s,
                            String from, String to) { ... }

    public static void main (String[] args) {
        System.out.println(norm("caravaggio"));
        System.out.println(norm("VERMEER"));

        System.out.println(init("johann sebastian bach"));
        System.out.println(init("i. babeL"));
        System.out.println(init("jorge LUIS BORGES"));
        System.out.println(init("WOLFGANG a. mozart"));

        System.out.println(tr("November 2016",
                    "abcdefghijklmnopqrstuvwyz",
                    "ABCDEFGHIJKLMNOPQRSTUVWYZ"));
        System.out.println(tr("abcXYZ","aZcX","||Cx"));
    }
}
```

should print

```
Caravaggio
Vermeer
J. S. Bach
I. Babel
J. L. Borges
W. A. Mozart
NOVEMBER 2016
|bCxY|
```

Do not use any classes except those in the package *java.lang*.

Problem 3

Class **Node** describes a node of singly-linked list and contains data of type **int** and the reference to the next node of the list.

Write a program operating on lists represented by references to objects of type **Node** which are their heads. Define the following functions:

1. `static Node arrayToList(int[] arr)`
   which takes an array of **int**s. The function creates the list of nodes containing integers from the array (in the same order) and returns the head of this list.
2. `static Node[] extract(Node head)`
   which takes the reference to the head of a list. The function splits the list into two — one containing only nodes with even data and the other with nodes containing odd data. Two-element array of heads of these lists is returned. NOTE: the **extract** function operates on existing nodes only; no new nodes are created!
3. `static void showList(Node head)`
   which prints the list represented by its head.

The following fragment:

```
int[] tab = { 2, 1, 4, 3, 6, 5, 7, 8 };
    Node head = arrayToList(tab);
    showList(head);
    Node[] nodes = extract(head);
    showList(nodes[0]);
    showList(nodes[1]);
```

should print something like

```
    2 1 4 3 6 5 7 8
    2 4 6 8
    1 3 5 7
```

Do **not** use any classes from the *java.util* package.
