Assignment 11

Topics: File and Folders, and Object Serialization

NOTE: exercises with prefix * are not mandatory**

1. Read about object serialization (mainly section 1 "System Architecture") at:
   http://docs.oracle.com/javase/7/docs/platform/serialization/spec/serialTOC.html

2. Write a class *FileInfo.java* that takes the path of a folder as parameter to the constructor of the class. Then add a method which counts and prints out how many files there are in the folder excluding subfolders (see classes *File* and *FileInfo*). Add another method that prints out the size of each file contained in the folder along with its type. Add a method that does the same, yet only for specific files according to a parameter passed as argument (e.g. if you pass *.doc, then only files of type doc should be considered). Add now a method that recursively counts the number of files in the folder and in all its subfolders.

3. Using the classes *ObjectOutputStreams* and *ObjectInputStreams* save a couple of *Student* objects from your previous assignments. Read them in again with an *ObjectInputStream*. Try to save an object with contain a reference to another object which does not implement the *Serializable* interface and explain what happens.
   Save now an object *Student* then remove its .class file. Try now to read in the again the object and explain what happens. Try now to modify the fields of the *Student* class e.g. by first changing a field name only, then by adding a new field, then by deleting a field, and so on. After each of these changes, save an object *Student* and read it in. Explain when you can do what. Now, use the class *Student* as you wish it and add the class constant *serialVersionUID* as:

   private static final long serialVersionUID = 1L;

   Make the changes as above and explain what happens and why.
   At this URL http://www.javablogging.com/what-is-serialversionuid/ you can read about the *serialVersionUID* constant.

4. *** Find out what changes to a class can hurt the deserialization process and what changes are instead usually ok

5. *** Create your own decorator for any I*nputStream* streams. Your decorator should turns all the characters specified in a variable of type *char[]* into either a single (default) character or into another set of characters defined in another variable of type *char[]*.

   Assuming that your decorator class is called *MyMaskForInputStream* then you need to define at least the two following constructors:

   MyMaskForInputStream(InputStream in, char[] mask, char c)
   MyMaskForInputStream(InputStream in, char[] fromSet, char[] toSet)

You must be able to decorate with your class all *InputStream* streams like in the following examples.

Example 1: decorating a *FileInputStream*

```
char[] fromSet = {'a','e','i','o','u'};
char toChar = '*';
InputStream in = new MaskInputStream(new FileInputStream("test.txt"), fromSet, toChar);
// here you print out the data in the Input stream
```

This code would replace all the occurrences of the characters *'a','e','i','o','u'* in file *test.txt* with the single star character *.

Example 2: decorating a *FileInputStream*

```
char[] fromSet = {'a','e'};
char[] toSet = {'o','u' };
InputStream in = new MaskInputStream(new FileInputStream("test.txt"), fromSet, toSet);
// here you print out the data in the Input stream
```

This code would replace all the occurrences of the characters *'a'* and *'e'* in file test.txt with the characters *'o'* and *'u'*, respectively. Notice that *fromSet* and *toSet* must have the same size!

Example 3: decorating the standard *InputStream*

```
char[] fromSet = {'a','e'};
char[] toSet = {'o','u' };
System.out.print("Enter a sentence here → ");
InputStream in = new MaskInputStream(System.in,fromSet,toSet);
// here you print out the data in the Input stream
```

This code would replace all the occurrences of the characters *'a'* and *'e'* that you type in with the keyboard with the characters *'o'* and *'u'*, respectively. Notice that *fromSet* and *toSet* must have the same size!