GET Parameter Encoder

You get key value pairs from an outside system, your job is to encode the values using the provided rules and print it like a URL GET parameters:

For example:

```
ParamEncoder encoder = // ...
// ...
SortedMap<String, Object> fieldValues = new TreeMap<String, Object>();
fieldValues.put(”name1”, ”value1”);
fieldValues.put(”name2”, ”value2”);
String result = encoder.encode(fieldValues);

// result is now ”name1=value1&name2=value2”
```

Assume that there is an interface ParamEncoder as follows

```
interface ParamEncoder {
        String encode (SortedMap<String, Object> data);
}
```

You need to provide a proper implementation for the above interface (the implementation details are mentioned further down).

Each field that needs to be encoded has a specified *maximum width*. When the value of that field exceeds this width, it should be truncated. Different fields have different truncation rules.

To capture these truncation rules, ParamEncoder has the method addFieldTruncationRule as:

```
interface ParamEncoder {
        // ...
        enum TruncationStyle { INTEGER, STRING_LEFT, STRING_RIGHT }
        void addruncationRule(String fieldName, TruncationStyle style, int maxWidth);
}
```

The enum values INTEGER, STRING_LEFT and STRING_RIGHT respectively represent the three styles of truncation outlined below.

1. Integer values should be *clamped* towards zero and is denoted by TruncationStyle.INTEGER. For example, consider an integer field x with maximum width 2 using this truncation style:

• A value of 1 will be encoded as x=1

• A value of 12 will be encoded as x=12

• A value of 123 will be encoded as x=99, since 99 is the largest positive number that can be encoded with 2 characters.

• A value of -1 will be encoded as x=-1

• A value of -12 will be encoded as x=-9, since -9 is the smallest negative number that can be encoded with 2 characters.

2. Some string values should be *truncated on the left* and is denoted by TruncationStyle.STRING_LEFT. For example, consider a string field x with maximum width 2 using this truncation style:

- A value of A will be encoded as x=A
- A value of AB will be encoded as x=AB
- A value of ABC will be encoded as x=BC, truncating on the left
- A value of ABCD will be encoded as x=CD, truncating on the left

3. Some string values should be *truncated on the right* and is denoted by TruncationStyle.STRING_RIGHT. For example, consider a string field x with maximum width 2 using this truncation style:

- A value of A will be encoded as x=A
- A value of AB will be encoded as x=AB
- A value of ABC will be encoded as x=AB, truncating on the right
- A value of ABCD will be encoded as x=AB, truncating on the right

Furthermore, some fields are arrays with repeating values. Encoding rules for such fields are captured with the method addArrayTruncationRule in ParamEncoder as:

```
interface ParamEncoder {
// ..
        void addArrayTruncationRule(String fieldName, int maxArrayWidth, TruncationStyle
elemStyle, intmaxElemWidth);
}
```

The arrays are encoded as comma-separated values enclosed by square brackets. Each element of the array is to be encoded in the given TruncationStyle, with no element taking more than maxElemWidth characters. The array as a whole, including the square brackets and the separator commas, should not take more than maxArrayWidth characters. For example, consider an array field x with maximum array width 10 using STRING_RIGHT truncation style and maximum element width 3:

- A value of ”AB”, ”CD” will be encoded as x=[AB,CD]
- A value of ”AB”, ”CDEF” will be encoded as x=[AB,CDE]; the second element got truncated in STRING_RIGHT style.
- A value of ”AB”, ”CDEF”, ”GHIJ”, ”K” will be encoded as x=[AB,CDE]; including the encoding of third element will make the encoded string longer than 10 characters, so it and all subsequent elements are dropped from the output.

Putting all these together, we get the ParamEncoder interface as:

```
import java.util.SortedMap;
interface ParamEncoder {
        enum TruncationStyle { INTEGER, STRING_LEFT, STRING_RIGHT }
        addFieldTruncationRule(String fieldName, TruncationStyle style, int maxWidth);
        addArrayTruncationRule(String fieldName, intmaxArrayWidth,
                TruncationStyle elemStyle, intmaxElemWidth);
        String encode(SortedMap<String, Object> data);
}
```

Here is some sample client code using ParamEncoder:

```
ParamEncoder encoder = new ParamEncoderImpl();
encoder.addFieldTruncationRule("left2", STRING_LEFT, 2); encoder.addFieldTruncationRule("left3",
STRING_LEFT, 3); encoder.addFieldTruncationRule("right2", STRING_RIGHT, 2);
encoder.addFieldTruncationRule("right3", STRING_RIGHT, 3); encoder.addFieldTruncationRule("int2",
INTEGER, 2); encoder.addFieldTruncationRule("int3", INTEGER, 3);
encoder.addArrayTruncationRule("array1", 10, STRING_RIGHT,3);
String result = encoder.encode(new TreeMap<String, Object>() {{
        put("array1", new String[] { "ABC", "EF", "IJ" });
        put("int2", 100);
        put("int3", 100);
        put("left2", "ABC");
        put("left3", "ABC");
        put("right2", "ABC");
        put("right3", "ABC");
        put("ignored","1234");
}});
//result is:
//"array1=[ABC,EF]&int2=99&int3=100&left2=BC&left3=ABC&right2=AB&right3=ABC";
```

Your assignment is to implement ParamEncoder along with appropriate unit tests. Your solution will be reviewed by the engineers you would be working with if you joined LeanTaaS. Here are our requirements:

- Please split the solution into multiple files, classes or methods - just as you would do in a professional setting. We are interested in seeing your real-world design, coding, documentation and testing skills.
- In the above description we have not specified what the expected behavior should be for many error conditions. Please use your judgement in such cases and document those decisions.

- Please try to get good coverage with your tests. We prefer unit tests to be written in one of JUnit or TestNG.
- Please deliver your solution in a tar.gz or zip archive file.