It Just Might Happen

File Format for Assignment 2

Example

“classes”:[

{

“className” : “exampleClass”,

“accessModifier” : “public”,

“isInterface” : false,

“fields” : [

{

“fieldName” : “exampleField”,

“fieldType” : “int”,

“accessModifier” : “public”,

“getter” : true,

“setter” : true

}

],

“functions”:[

{

“functionName” : “exampleFunction”,

“functionType” : “int”,

“security” : “public”,

“parameters” : [],

“constructor” : “default”

}

]

}

]

# General Information

The file format we will be using for this project is JSON, or Javascript Object Notation. This is a simple, readable, writable format that is also extremely capable. JSON is a way of storing data as fields, objects, or array. Whitespace does not matter, one does not need to format the JSON in any particular way, beyond the basics. The entire file will be contained within two square brackets, making an array. Within this array will be class descriptions enclosed in curly brackets. These class descriptions will contain fields, which are formatted with field names being enclosed in double quotes, followed by a colon, and the field value in double quotes. These fields are delineated by commas.

# General Class Fields

### “className”

This field will contain the class name. The potential values for “className” are any string that is a legal class name in the language you will be using. For example, in Java “getValue” is a legal name, whereas “int” is not.

### “security”

This field will contain the security status of the class. The potential values for “security” are “public”, “private”, and “protected”.

### “isInterface”

This field will contain a Boolean that states whether the class is an interface or not.

### “fields”

This field will contain an array of fields, of length zero or more, see Field Fields.

### “functions”

This field will contain an array of functions, of length zero or more, see Function Fields

# Field Fields

### “fieldName”

This field will contain the name of the field. The potential values for “fieldName” are any string that is a legal field name. The same example applies, “getValue” is a legal name, “int” is not.

### “fieldType”

This field will contain the type of the field. Definitely supported types are “int”,”short”, ”long”, ”double”, “float”, “string”, “char”, and “boolean”. Data types that may be supported are arrays of the supported data types.

### “security”

This field will contain the security status of the field. The potential values are “public”, “private”, and “protected”.

### “value”

This field will contain the value of the field. Potential values are those which are allowed within the data type the field is, for example, “5” is acceptable for a integer, whereas “5.5”, or “Hello World” are not.

# Function Fields

### “functionName”

This field contains the function name. Potential values are any legal function name in the language you are using.

### “functionType”

This field will contain the function type. Supported data types are “int”, “short”, “long”, “double”, “string”, “char”, and “boolean”.

### “security”

This field will contain the security status of the function. The potential values are “public”, “private”, and “protected”.

### “parameters”

This field will contain an array of fields that are the parameters of the function. See Field Fields.

### “constructor”

This field will give the appropriate information for the generation of constructors. Potential values are an array, which either contains “null”, “default”, or a list of names of fields that have been provided in the “fields” array. These field names should have the same index as the value in parameters that they will be set to, for example, if one wants to set the string message to “Hello World”, constructor[i] should be “message”, and paremeters[i] should be “Hello World”.

### “setter”

This field will give the appropriate information for the generation of setters. Potential values are an array, which either contains “null” or a list of names of fields that have been provided in the “fields” array. These field names should have the same index as the value in parameters that they will be set to, for example, if one wants to set the string message to “Hello World”, constructor[i] should be “message”, and paremeters[i] should be “Hello World”.

### “getter”

This field will give the appropriate information for the generation of getters. Potential values are “null” and the name of a single field described in the fields array. If this value is not set to none, only one parameter will be accepted.