UNIVERSITÄT DUISBURG ESSEN

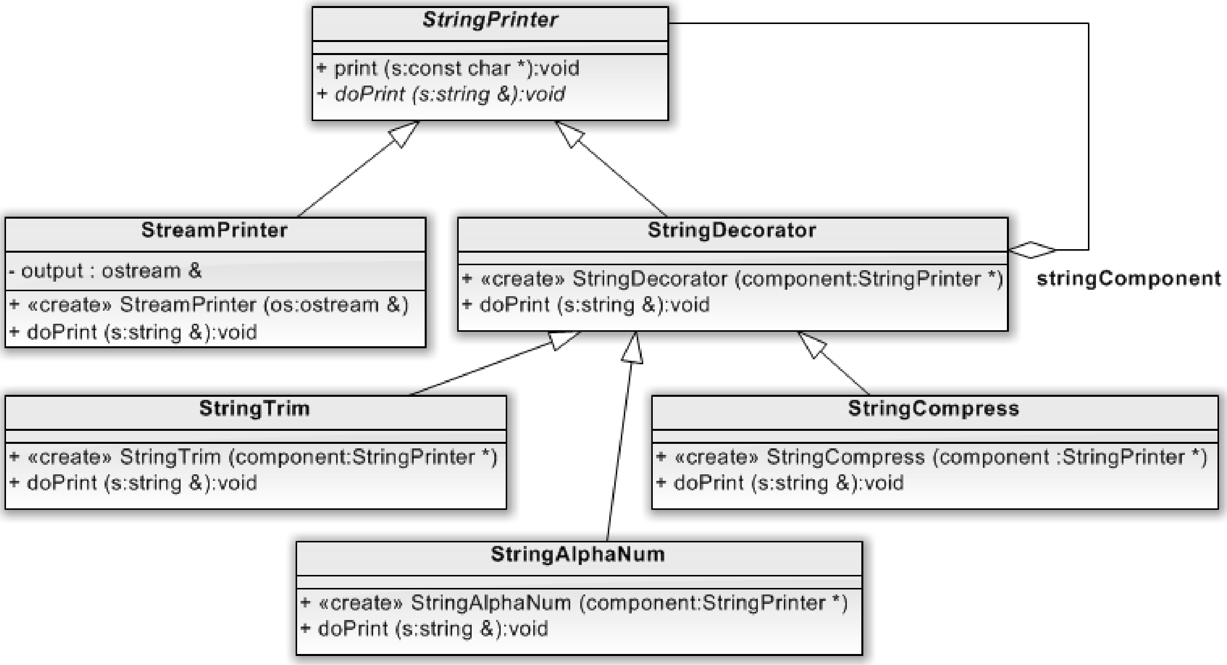
NES Networked Embedded Systems

Programming in C/C++

Exercises 8

Deadline: 14.01.2015 11:00

1. Problems in software engineering that occur frequently can be solved using so-called design patterns. The “Decorator” pattern is such a design pattern that allows for extending the functionality of a class in a flexible way. In this task you implement a class hierarchy that prints out a string that can be modified in different ways before printing. The class diagram is shown here:


StringPrinter
+ print (s:const char *):void
+ doPrint (s:string &):void
StreamPrinter
- output : ostream &
+ «create» StreamPrinter (os:ostream &)
+ doPrint (s:string &):void
StringDecorator
+ «create» StringDecorator (component:StringPrinter *)
+ doPrint (s:string &):void
stringComponent
StringTrim
+ «create» StringTrim (component:StringPrinter *)
+ doPrint (s:string &):void
StringCompress
+ «create» StringCompress (component :StringPrinter *)
+ doPrint (s:string &):void
StringAlphaNum
+ «create» StringAlphaNum (component:StringPrinter *)
+ doPrint (s:string &):void

- Class “`StringPrinter`” provides the method `print()` that is called with a C string and creates a std::string from it. Then it calls `doPrint()` with this string. This allows for a modification of the string without copying it in every step. `doPrint()` shall be a pure virtual function.
- Class “`StreamPrinter`” is a concrete implementation of a `StringPrinter`. Its constructor accepts an output stream that is stored internally. In `doPrint()`, string s is output to this stream, with a ‘|’ before and after the string. (Remark: The general idea is to have more than one concrete implementation of `StringPrinter` that all work with the same decorator classes, e.g. a printer that outputs to a graphical window. We are using only one type here.)
- Class “`StringDecorator`” is the super class for all string modifiers (“decorators”). It manages a pointer to a StringPrinter object, stored by the constructor. Its `doPrint()` method simply calls `doPrint()` of this object.
- Classes “`StringTrim`”, “`StringAlphaNum`” and “`StringCompress`” are three concrete decorators. “`StringTrim`” removes all whitespaces at the beginning and at

the end of the string, “StringAlphaNum” replaces all non-alphanumeric characters (0-9,a-z,A-Z) with spaces, and “StringCompress” compresses two or more consecutive spaces to a single space. (Hint: Look at isspace() and isalnum() in the cctype library and the string functions of the string library. http://www.cplusplus.com/reference/ is, for example, a good reference. You can also use iterators if you want to).

Test your code with the provided decorator.cpp test program. Put your class definitions and implementations all in the decorator.cpp file, just before main()

(code 60 pts, comments 10 pts)

2. The given stack.cpp implements a static int-stack. Transform it to a template class with two arguments, one for the data type and one for the size. “Stack<double,10> s;” then defines s to be a stack of 10 doubles.

Instead of simply printing out errors, push() and pop() shall throw exceptions in case of errors. For this purpose, add a class StackException that stores a reason (e.g. as C string) why the exception occurred.

Change the main program in such a way that the template class is used now and that the exceptions are caught and their reason is printed. STACKSIZE must not appear in the program any more.

Maintain the structure of the stack.cpp file, i.e. keep your class definitions separate from the implementation but all in the same file.

(code 30 pts)