z/VM GUI Project
Coding Standards Guide
Version 0.1 09/27/04

Overview

The purpose of this document is to set forth a standard template for creating code that will be contributed to the Clarkson Open Source Institute z/VM GUI project. All code contributed by members of the organization should follow all standards as closely as possible.

Basic Java Source Code File Structure

All Java source code files should contain the following code sections. If a section does not apply to a file (for example, if the code does not need to use import statements) then that section should simply be omitted without any type message, signal, or place holder. Some sections are always required and are marked as such.

- Main comment (required)
- Package statement (required)
- Import statements
- Class Java Doc (required)
- Main Class Definition (required)
  - Public data members
  - Private data members
  - Constructor(s)
  - Public function definitions
  - Private function definitions

Main Comment

The main comment is the project/license/author information that should appear at the beginning of every source file. At the time of this documents creation, the main comment in use is:

```
/*
 * Author: Author Name (Author Email)
 *
 * The project that this code belongs to has a web page at
 * http://www.clarkson.edu/projects/cosi/zTeam/zvmgui/
 *
 * This version is licensed under the OSL (Open Software
 * License) v. 2.0. The OSL is found in LICENSE.TXT distributed
 * with this code. The original copy of the OSL may be found at
 * http://www.opensource.org/licenses/osl-2.0.php
 *
 * We will consider licensing this code under other licenses.
 * Send email to the director of COSI if you are interested.
 *
 * The z/VM GUI Project is OSI Certified Open Source Software.
 *
 * Copyright 2004, Clarkson Open Source Institute
 * (http://cosi.clarkson.edu)
 */
```

Package statement

The package statement should consist of the word package, followed by the package that the file belongs to and a semicolon. One single blank line should separate the main comment from the package statement.

Example:

```
package zvmgui.communication;
```

Import Statements

All import statements should be specified in this section. Standard java library imports should go first, followed by project specific imports. A single blank line should be used to separate import statements if you wish to group them. In any case, a single blank line should at least separate the standard java library imports from the project specific imports.

Example without grouping:

```
import java.io.*;   //InputStream, BufferedInputSteam,
OutputStream
import java.nio.*;      // for byte-to-char decoding
import java.nio.charset.*; // for byte-to-char decoding
import java.util.Arrays; // for byte-to-char decoding
import java.util.StringTokenizer;

import zvmgui.exceptions.*;
import zvmgui.exceptions.clc.*;
import zvmgui.exceptions.telnet.*;
import zvmgui.communication.CommandTextProcessor;
import java.util.LinkedList;
```

Example with grouping:

```
import java.io.*;   //InputStream, OutputStream
import java.nio.*;      // for byte-to-char decoding
import java.nio.charset.*; // for byte-to-char decoding

import java.util.Arrays; // for byte-to-char decoding
import java.util.StringTokenizer;

import zvmgui.exceptions.*;
import zvmgui.exceptions.clc.*;
import zvmgui.exceptions.telnet.*;

import zvmgui.communication.CommandTextProcessor;
import java.util.LinkedList;
```

Class Java Doc

One or more blank lines should separate the import statements from the class Java Doc.

A Java Doc comment that describes the class should appear here. Feel free to be verbose in this description as it will server as the main documentation for this class. This Java Doc comment should always contain an @author field that should list the current author/maintainer.

Example:

```
/**
 * Provides an interface that allows for execution of z/VM
 * commands and
 * retreival of output caused by execution of those commands.
 *
 * @author Jason J. Herne (hernejj@clarkson.edu)
 */
```

Main Class Definition

The main class definition should immediately follow the class Java Doc comment. no blank lines should be used to separate them.

This is where the definition and the code for the class goes.

Inside the Main Class

Inside, the main class, public data members should precede private data members. Functions should come next. Constructors first, then public functions and then private functions.

Public data members

A single blank line (and/or comments) may separate the end of the class opening brace from the public data members.

All public data members should explicitly use the keyword “public”. Any constants (public or private) should precede the normal public data members.

Example:

```
//Constansts
public final byte ZVM_MESSAGE_TERMINATOR = 0x11;
public final byte ZVM_NEWLINE_1of2 = 0xD; // \r
public final byte ZVM_NEWLINE_2of2 = 0x00; // \0

//Telnet session used.
public TelnetController telnetController;
```

Private data members

A single blank line should separate public data members from private data members. A comment may be used as well.

Example:

```
// I/O streams.
private BufferedInputStream is;
private OutputStream os;
```

Constructor(s)

All constructors should precede the rest of the functions in the class definition.

If there is a default constructor (a constructor that takes no arguments) it should precede other constructors. Ideally, constructors should proceed in the order of simplest (at the top) to most complex.

All constructors need corresponding Java Doc comment blocks placed directly above them. All parameters need to have a Java Doc @parm statement.

Example:

```
/**
  * Creates a new zVMCommandLineController and connects to host
  * at port  23.
  * @parm host The host to establish a z/VM connection with.
  */
public zVMCommandLineController(String host)
      throws telnetCantConnectException,
{
      //...
}

/**
 * Creates new zVMCommandLineController,connects to host at myPort
 *
 * @parm host The host to establish a z/VM connection with.
 * @parm port The port on which to connect to z/VM.
 */
public zVMCommandLineController(String host, int myPort)
      throws telnetCantConnectException
{
      //...
}
```

Public function definitions

Public functions are next. They should always explicitly use the keyword “public”. They also must have Java Doc comments directly above them. All parameters need to have a Java Doc @parm statement, and any non-void function must have a Java Doc @return statement.

Private function definitions

Private functions should be placed after public functions and should follow all of the rules that public functions do.

Naming Conventions

All variables (unless constants) should begin with a lowercase letter. Individual words in variable names should be capitalized.

Example:

```
telnetController
```

All constant variables should contain only upercase characters, underscores and numbers.

Example:

```
ZVM_MESSAGE_TERMINATOR
```

All class definitions should start with an uppercase letter and Individual words within class names should be capitalized.

Example:

```
GuestLan
```

Indenting Style

All braces should be placed on separate lines. The K&R style of bracing will not be used.

A single tab shall be used to indent code between braces. All beginning and ending braces should line up.

In the event that a closing brace appears a great distance from its opening brace, a comment can be used to clarify it's purpose.

Example:

```
public class GuestLan
{
	public void print()
	{
		while(true)
		{
			if(...)
			{
				//
			}
			else
			{
				//...
			}
		}
	} // end: print()
} //end: class GuestLan
```